

SOFTWARE QUALITY ASSESSMENT USING ENSEMBLE MODELS

BY

HAMOUD IBRAHIM ALJAMAAN

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

June 2009


KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS


DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Hamoud Ibrahim Aljamaan** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**

Thesis Committee


Dr. Mahmoud Elish (Thesis Advisor)


Dr. Muhammed Al-Mulhem (Member)


Dr. Mohammad Alshayeb (Member)


Dr. Kanaan A. Faisal
Department Chairman


Dr. Salam A. Zummo
Dean of Graduate Studies


Date

DEDICATION

To my father and mother

To my wife

To my brother and sisters

ACKNOWLEDGMENT

All praise is due to Allah the Almighty for his countless blessings and enlightenments throughout my studies.

I would like to express my gratitude to my advisor, Dr. Mahmoud Elish, for his wise guidance and motivation. I really enjoyed working with him, and I think he sets an example of what an advisor should be. My gratitude is extended to my committee members, Dr. Muhammed Al-Mulhem and Dr. Mohammad Alshayeb, for their continuous support and encouragement. Their valuable feedback improved the content of this research. Furthermore, I would like to thank my friends and colleagues, who we shared knowledge throughout my graduate studies.

I would like to acknowledge the support of King Fahd University of Petroleum and Minerals in the development of this research. Also, special thanks are due to King Abdulaziz City for Science and Technology (KACST) for funding this research. This research is done under project number GSP-17-132.

Sincere admiration is for my father Dr. Ibrahim Aljamaan, and I am proud to be your son. My mother is my constant refuge, and I am proud to share my life with my wife.

TABLE OF CONTENTS

DEDICATION	I
ACKNOWLEDGMENT	IV
TABLE OF CONTENTS.....	V
LIST OF TABLES	IX
THESIS ABSTRACT.....	XIII
ملخص الرسالة	XIV
CHAPTER 1	1
INTRODUCTION	1
1.1 Problem statement	1
1.2 Software engineering prediction problems	3
1.3 Motivation	5
1.4 Thesis objective.....	7
1.5 Thesis organization	7
CHAPTER 2	8
LITERATURE REVIEW	8
2.1 Object-Oriented design metrics as quality indicators	8
2.1.1 Relationships between object-oriented design metrics and class fault proneness	9
2.1.2 Relationships between object-oriented design metrics and maintainability	10
2.2 Class fault prediction.....	11
2.3 Maintainability prediction.....	13

2.4	Ensemble models	14
2.5	Summary.....	15
CHAPTER 3		18
TECHNICAL BACKGROUND.....		18
3.1	Individual prediction models.....	18
3.1.1	Artificial Neural Networks.....	19
3.1.1.1	Multilayer Perceptron (MLP).....	21
3.1.1.2	Radial Basis Function Network (RBF).....	24
3.1.2	Statistical.....	25
3.1.2.1	Bayesian Belief Network (BBN)	26
3.1.2.2	Naïve Bayes (NB)	27
3.1.3	Support Vector Machines	28
3.1.3.1	Support Vector Machines (SVM).....	28
3.1.3.2	Support vector regression (SVMreg)	31
3.1.4	Tree.....	32
3.1.4.1	Decision Tree (DT).....	32
3.1.4.2	M5 Model tree (M5P)	33
3.1.5	Parameter initialization.....	35
3.2	Datasets	36
3.2.1	Class fault datasets.....	37
3.2.2	Maintainability datasets	39
3.3	Tool.....	41
3.4	Evaluation measures.....	42
3.4.1	Classification evaluation measures.....	42
3.4.1.1	Accuracy.....	43
3.4.1.2	Recall	43
3.4.1.3	Precision.....	43
3.4.1.4	F- measure	43
3.4.1.5	ROC.....	44
3.4.2	Regression evaluation measures.....	45
3.4.2.1	MMRE.....	46
3.4.2.2	StdMRE.....	46
3.4.2.3	Pred(0.3)	46
3.5	Validation techniques	47
3.5.1	Cross validation	47
3.5.1.1	10 fold cross validation	47
3.5.1.2	Leave-one-out	48
3.5.2	Holdout method	48

CHAPTER 4	49
SINGLE MODEL ENSEMBLES (CLASSIFICATION).....	49
4.1 Single model ensembles for classification	49
4.1.1 Bagging.....	51
4.1.2 Boosting	52
4.2 Experiment design.....	54
4.2.1 Goal	54
4.2.2 Settings.....	54
4.2.3 Bagging and boosting parameters	55
4.3 Results	55
CHAPTER 5	62
MULTI-MODEL ENSEMBLES (CLASSIFICATION)	62
5.1 Multi-model ensembles for classification	62
5.1.1 Linear ensembles.....	62
5.1.1.1 Majority voting	64
5.1.1.2 Average probability	64
5.1.1.3 Best probability	65
5.1.1.4 Weighted probability	66
5.1.2 Nonlinear ensembles	68
5.2 Experiment design.....	70
5.2.1 Goal	70
5.2.2 Settings.....	71
5.3 Results	71
CHAPTER 6	81
MULTI-MODEL ENSEMBLES (REGRESSION)	81
6.1 Multi-model ensembles for regression	81
6.1.1 Linear ensembles.....	81
6.1.1.1 Average	83
6.1.1.2 Best.....	83
6.1.1.3 Weight.....	84
6.1.2 Nonlinear ensembles	85

6.2	Experiment design.....	87
6.2.1	Goal	87
6.2.2	Settings.....	88
6.3	Results	88
6.3.1	KC1.....	88
6.3.2	UIMS.....	96
6.3.3	QUES.....	103
CHAPTER 7		111
CONCLUSION		111
7.1	Thesis contributions.....	111
7.2	Limitations	113
7.3	Future work	114
REFERENCES		115
APPENDIX A – DETAILED EXPERIMENT RESULTS.....		127
VITA.....		170

LIST OF TABLES

Table 1: Literature review summary	17
Table 2: Selected individual prediction models	19
Table 3: Software quality datasets.....	37
Table 4: KC1 independent and dependent variables.....	39
Table 5: UIMS & QUES independent and dependent variables.....	41
Table 6: Confusion matrix for binary classification problems.....	42
Table 7: Comparison between bagging and boosting.....	54
Table 8: Experiment settings for single classifier ensemble (classification)	54
Table 9: Classification accuracy of single classifiers.....	56
Table 10: Classification accuracy after applying bagging on base classifiers using different ensemble size.....	57
Table 11: Classification accuracy after applying boosting on base classifiers using different ensemble size.....	57
Table 12: Experiment settings for multi-model ensemble (classification)	71
Table 13: KC1 results for individual models (classification)	72
Table 14: Individual models AUC (KC1 classification)	74
Table 15: KC1 results for linear ensembles (classification)	75
Table 16: Linear ensemble models AUC (KC1 classification).....	76
Table 17: KC1 results for nonlinear ensembles (classification).....	77
Table 18: Nonlinear ensemble models AUC (KC1 classification).....	78
Table 19: KC1 comparison of individual Vs best linear Vs best nonlinear (classification).....	79
Table 20: Comparison of individual Vs best linear Vs best nonlinear AUC (classification)	80
Table 21: Experiment settings for multi-model ensemble (regression)	88
Table 22: KC1 results for individual models (regression)	89
Table 23: KC1 results for linear ensembles (regression)	91
Table 24: KC1 results for nonlinear ensembles (regression).....	93
Table 25: KC1 comparison of individual Vs best linear Vs best nonlinear (regression).....	94
Table 26: Wilcoxon MRE significance test of KC1 individual, best linear, and best nonlinear (p-level = 0.1)	95
Table 27: UIMS results for individual models (regression).....	96
Table 28: UIMS results for linear ensembles (regression).....	98
Table 29: UIMS results for nonlinear ensembles (regression)	100
Table 30: UIMS comparison of individual Vs best linear Vs best nonlinear (regression).....	101
Table 31: Wilcoxon MRE significance test of UIMS individual, best linear, and best nonlinear (p-level = 0.1)	102
Table 32: QUES results for individual models (regression)	104
Table 33: QUES results for linear ensembles (regression).....	105
Table 34: QUES results for nonlinear ensembles (regression).....	107

Table 35: QUES comparison of individual Vs best linear Vs best nonlinear (regression).....	109
Table 36: Wilcoxon MRE significance test of QUES individual, best linear, and best nonlinear (p-level = 0.1)	109

LIST OF FIGURES

Figure 1: Software Quality Assessment	2
Figure 2: Software errors, faults and failures.....	4
Figure 3: Artificial neuron [2].....	20
Figure 4: Feed-forward artificial neural network architecture [2].....	21
Figure 5: Example of a multilayer perceptron network [102].....	22
Figure 6: MLP neuron [102]	23
Figure 7: Radial basis function network [103].....	25
Figure 8: Example of Bayesian Belief Network.....	26
Figure 9: Maximal margin hyperplane	29
Figure 10: Data transformation into a non-linear space	31
Figure 11: A decision tree [87].....	33
Figure 12: M5 Model tree [8]	34
Figure 13: A ROC graph	44
Figure 14: Single classifier ensemble	50
Figure 15: Bagging algorithm.....	51
Figure 16: Boosting algorithm	52
Figure 17: Single classifier Vs Bagging Vs Boosting.....	58
Figure 18: (a) Bagging all classifiers, (b) Boosting all classifiers	59
Figure 19: Single classifier Vs Bagging (25) Vs Boosting (25).....	61
Figure 20: Multi-model linear ensemble (classification)	63
Figure 21: Majority voting linear ensemble.....	64
Figure 22: Average probability linear ensemble	65
Figure 23: Best probability linear ensemble.....	66
Figure 24: Weight probability linear ensemble.....	67
Figure 25: Multi-model nonlinear ensemble (classification).....	69
Figure 26: Nonlinear ensemble	70
Figure 27: KC1 ROC graph for individual models (classification)	73
Figure 28: KC1 ROC graph for linear ensembles (classification).....	76
Figure 29: KC1 ROC graph for nonlinear ensembles (classification).....	78
Figure 30: KC1 ROC graph for comparison of individual Vs best linear Vs best nonlinear (classification)	80
Figure 31: Multi-model linear ensemble (regression)	82
Figure 32: Average linear ensemble.....	83
Figure 33: Best linear ensemble	84
Figure 34: Weight linear ensemble	85
Figure 35: Multi-model nonlinear ensemble (regression).....	86
Figure 36: Nonlinear ensemble	87

Figure 37: KC1 box plots for individual models (regression)	90
Figure 38: KC1 box plots for linear ensembles (regression)	92
Figure 39: KC1 box plots for nonlinear ensembles (regression)	93
Figure 40: KC1 box plots of best individual Vs best linear Vs best nonlinear (regression)	96
Figure 41: UIMS box plots for individual models (regression)	97
Figure 42: UIMS box plots for linear ensembles (regression)	99
Figure 43: UIMS box plots for nonlinear ensembles (regression)	100
Figure 44: UIMS box plots of best individual Vs best linear Vs best nonlinear (regression)	103
Figure 45: QUES box plots for individual models (regression)	104
Figure 46: QUES box plots for linear ensembles (regression)	106
Figure 47: QUES box plots for nonlinear ensembles (regression)	108
Figure 48: QUES box plots of best individual Vs best linear Vs best nonlinear (regression)	110

THESIS ABSTRACT

Name: Hamoud Ibrahim Aljamaan
Title: Software Quality Assessment using Ensemble Models
Major Field: Information & Computer Science
Date of Degree: June 2009

During the different phases of a software project, the manager is faced with many prediction problems, particularly the software development effort and quality. Early prediction of quality helps the management to have the knowledge of targeted software product quality as early as possible, which helps to identify design errors and avoid expensive rework.

Many predication models have been proposed in the research community in order to achieve accuracy in software engineering related prediction problems. However, none of the existing prediction models proved to be suitable under most circumstances.

The main objective of this thesis is to build different ensemble models, and evaluate their accuracy against stand-alone prediction models. Several linear and nonlinear ensembles were proposed, and three empirical studies were conducted to evaluate them in the context of fault and maintenance effort prediction. Empirical results indicate that ensembles in general offer better, or at least competitive, performance by comparison with individual models, and nonlinear ensembles were the best among ensembles.

ملخص الرسالة

الاسم : حمود ابراهيم الجمعان

عنوان الرسالة: تقييم جودة البرمجيات باستخدام نماذج المجموعة "انسامبل"

التخصص: علوم الحاسب الآلي و المعلومات

تاريخ الشهادة: جمادى الآخرة 1430هـ

خلال المراحل المختلفة من مشروع البرمجيات، مدير مشروع البرمجيات يواجه العديد من المشاكل مثل التنبؤ بتطور البرمجيات ونوعية الجودة المستقبلية للبرمجيات. التنبؤ المبكر لجودة البرمجيات تساعد الإدارة على تكوين معرفة لنوعية جودة المنتجات المستهدفة في أقرب وقت ممكن، مما يساعد على تحديد وتصميم وتجنب الأخطاء المكلفة لإعادة صياغة البرمجيات. في مجتمع الأبحاث، هناك العديد من التقنيات المقترحة من أجل بناء نماذج للتنبؤ الدقيق لهندسة البرمجيات ذات الصلة بمشاكل التنبؤ، إلا أن أيًا من نماذج التنبؤ القائمة أثبتت أنها مناسبة في ظل معظم الظروف. الهدف الرئيسي لهذه الأطروحة هو بناء مجموعة نماذج "انسامبل" مختلفة، وتقييم دقة تنبؤها مقارنة بالتقنيات الفردية الموجودة. اقترحنا مجموعة من نماذج "انسامبل" الخطية وغير الخطية، وقد أجريت ثلاث دراسات تجريبية لتقييمها في سياق التنبؤ بأخطاء وصيانة البرامج. النتائج التجريبية تشير أن نماذج "انسامبل" تقدم بشكل عام أداء أفضل، أو على الأقل منافس لأداء التقنيات الفردية الموجودة، إضافة لذلك أثبتت نماذج "انسامبل" غير الخطية هي أفضل بين نماذج "انسامبل".

CHAPTER 1

Introduction

Software project management is the art of making the right decision [98] among a huge number of choices during the different phases of a project. The success and failure of projects is highly dependent on the manager's decisions. Managers should be encouraged to use any means necessary to help them make their decisions as accurate as possible to increase the overall project success rate.

Software project management is a major application area for prediction models. During the different phases of a project, the manager is faced with many prediction problems such as predicting the development effort, cost and quality. Prediction models can be used to help and guide software project managers to make right estimates when a prediction problem is faced. Software project managers can utilize such prediction models besides their work experience to come up with decisions that increase the overall project success.

1.1 Problem statement

Software quality assessment can be seen as a learning concept [3], where we can utilize prediction models as an application to it. Assessing software quality, as shown below, involves predicting several software attributes, such as maintainability, reliability, fault-proneness ... etc.



Figure 1: Software Quality Assessment

An important software quality attribute is maintainability. Nowadays, many software systems are currently in use and the largest cost associated with any product over its lifetime is the maintenance cost [66, 108]. Fault-proneness is another important aspect of quality. Faults are defined as defects that might cause failures. Predicting these prone parts is a challenge for developers before the software is released [56].

Building accurate prediction models helps software project managers increase the success of their software project. However, none of the existing prediction models proved to be suitable under most circumstances. In fact, for a given dataset one model may outperform other models. But, when a different dataset is used, the model used could produce the worst prediction accuracy. Ensemble models were proposed as a candidate approach to build an accurate prediction model by taking advantage of stand-alone prediction models capabilities towards the dataset to come up with the best, or at least competitive, prediction accuracy.

1.2 Software engineering prediction problems

A software project manager may face many prediction problems during the different phases of a project. Such predictions include:

- **Software Quality**

Predicting software quality involves predicting maintainability, fault-proneness, etc...

- **Software maintainability**

Many definitions exist for maintainability. Pressman defined maintainability as “the ease with which repair may be made” [82]. Also, following IEEE Standard Glossary of Software Engineering Terminology, maintainability is defined as “The ease with which a system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment” [52].

Maintainability is typically measured as change effort [62, 95]. Change effort can mean either the average effort to make a change to a class, or the total effort spent on changing a class [62, 95], and this is how software maintainability is addressed in this thesis.

- **Fault-proneness**

Software fault is defined as a defect (e.g. programmer error) in an executable product that may cause a failure [41, 52]. However, not all faults result in a failure, as shown in the Figure 2 [41]. A fault becomes a failure, once it is activated.

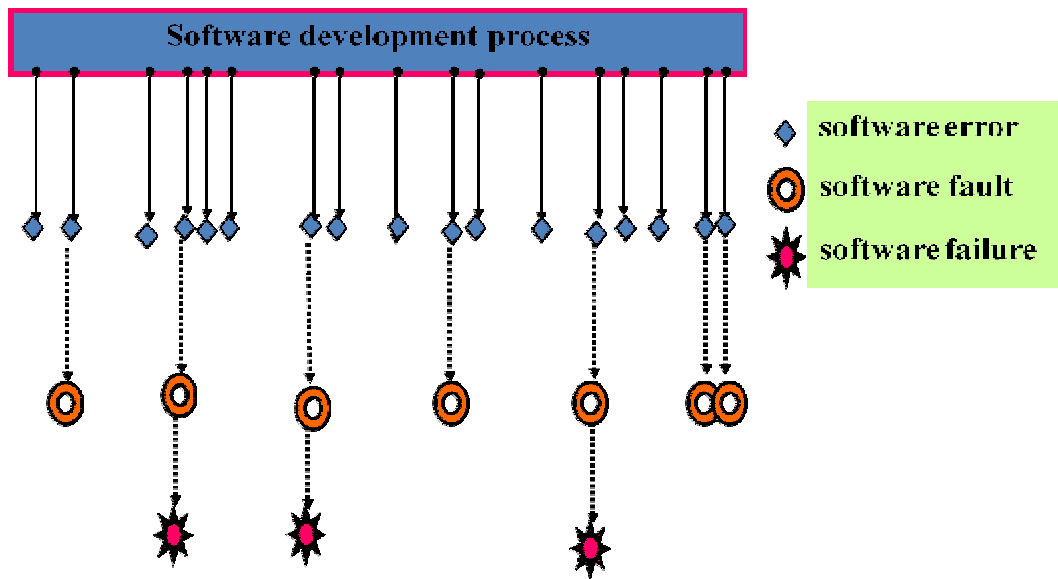


Figure 2: Software errors, faults and failures

Fault prediction can be in many forms. Examples include predicting faulty parts, before its release, or it may involve predicting the fault density of faulty parts... etc

- **Software reliability**

Reliability, according to ANSI standards, is defined as “the probability of failure-free operation for a specified period of time in a specified environment” [73]. Reliability is another factor for quantitatively characterizing quality and estimating the duration of testing period.

- **Software Cost**

Cost estimation may be used to establish a budget for the project. Cost usually includes effort (i.e. the cost of labor) and schedule (i.e. the cost of time), and therefore, we can define cost estimation as an empirical process of estimating the effort and time costs for

any product before developing it [11]. Thus, the costs of development are primarily the costs of the effort and time involved.

The primary topic and scope of this thesis is software quality, and specifically maintainability, and fault proneness at the class level. We approach the quality assessment by estimating the fault proneness and maintainability, since it provides us with assurances about quality [95]. Estimating class fault proneness is used to assess and shed the light on overall quality. If we can identify the important drivers of fault proneness, we can use them as candidates for quality benchmarks [7, 19, 28]. In addition, the more faults are found, the better the quality and maintainability of the system [104]. Also, quality can be assessed by estimating the maintainability (e.g. the number of lines changed per class) [72, 95, 108].

1.3 Motivation

Early prediction of software quality helps the management to have the knowledge of targeted product quality as early as possible, which helps to identify design errors and avoid expensive rework. Quality prediction models can be utilized as a tool for focusing quality assurance activities.

Software maintainability prediction is an important quality prediction problem. Many systems are currently in use and the largest cost associated with any product over its lifetime is the maintenance cost, thus, it is important for those systems to be maintained effectively and efficiently [5]. Maintainability prediction model enables organizations to predict maintainability of a system and assists them with managing maintenance resources. Accurate maintainability prediction enables developers to better

identify the determinants of quality and thus help them improve design or code, also it can provide project managers with useful information to help them plan the use of valuable resources [62]. In addition, if an accurate maintainability prediction model is available for a software system, a defensive design can be adopted. This would minimize, or at least reduce future maintenance effort of the system [67].

Software fault prediction is another important quality prediction problem. The majority of faults in a software system are found in a few of its components [12, 81]. Thus, early prediction of faults is a challenging task for the developers before the release [56]. Accurate fault prediction models enables developers to focus quality assurance activities and allocate effort and resources more efficiently [47]. Therefore, predicting fault-proneness is important for minimizing cost and improving the effectiveness of the testing process. As a result, accurate prediction models can lead to a substantial improvement in quality [61].

Building quality prediction models that are accurate and suitable under most circumstances is critical for the overall project success. Ensemble models are a candidate approach for this goal. They take advantage of stand-alone prediction models capabilities towards the dataset to come up with competitive prediction accuracy.

Our hypothesis is that ensemble models will help project managers to make more accurate predictions. Once supported, project managers can utilize ensemble models besides their work experience to come up with decisions that increase the overall project success.

1.4 Thesis objective

The main objective of this thesis is to build different ensemble models, and evaluate their prediction accuracy against stand-alone prediction models for fault and maintainability prediction. To accomplish this, we carried out the following tasks:

- Survey existing prediction models and identify the most commonly used models.
- Propose different linear and nonlinear ensembles from existing stand-alone prediction models.
- Search for different datasets for software engineering quality prediction problems.
- Empirically validate the ensemble models with respect to their prediction accuracy over other stand-alone prediction models.

1.5 Thesis organization

The remaining of this thesis is organized as follows: Chapter 2 reviews the related work done in the field of software fault and maintainability prediction. Chapter 3 provides the common technical background for all conducted experiments. Chapter 4 describes building ensembles from a single model, and presents the empirical study conducted to evaluate them. Chapter 5 and 6 describe building ensembles using different models for classification and regression problems respectively, and then present the empirical studies conducted to evaluate them. Chapter 7 discusses thesis conclusions, and provides directions for future work.

CHAPTER 2

Literature review

This chapter reviews the related work done in the area of software fault and maintainability prediction. It shows the use of object oriented metrics as quality indicators, work done in the area of ensemble models, and the related work done in predicting maintainability and class fault proneness.

2.1 Object-Oriented design metrics as quality indicators

Nowadays object oriented (OO) approach to software development is widely used in the software industry. OO promises better management of system complexity and improvement in project outcomes such as software quality. Thus, a wide variety of OO design metrics have been proposed to assess the quality of an OO system [9, 16, 20, 22, 50, 55, 65, 66].

The most widely used OO design metrics suite is Chidamber and Kemerer's suite [22]. This suite mainly focuses on classes and can capture different aspects of an OO design. It helps to identify areas of the application that may require more rigorous testing and areas that are candidates for redesign. In addition, it helps in predicting certain project outcomes and external software qualities such as software faults.

2.1.1 Relationships between object-oriented design metrics and class fault proneness

Recent research studies validated the use of OO design metrics as quality measurement of an OO system. By relating metrics to fault-proneness, we can identify the important drivers of fault-proneness. Thus, OO design metrics can be used as early quality indicators and detectors of faulty classes.

Many researchers investigated the use of OO design metrics as early quality indicators and tried to define suitable metrics for fault detection. Their studies provided empirical evidence that a correlation exists between some software metrics and class fault-proneness [7, 18, 19, 28, 34, 93, 104].

Briand et al. [19] investigated the relationship between existing design measurement in OO systems and the quality of the software developed. Univariate analysis results have shown that many coupling and inheritance measures are strongly related to the probability of fault detection in a class. However, cohesion does not seem to have a significant impact on fault proneness.

Basili et al. [7] assessed the use of OO design metrics as predictors of fault-prone classes. Based on empirical and quantitative analysis, their results showed that five out of the six Chidamber and Kemerer's OO metrics appear to be useful in predicting class fault-proneness during the high- and low-level design phases of the life-cycle. In addition, Chidamber and Kemerer's OO metrics were shown to be better predictors than the best set of traditional code metrics, which can only be collected during later phases of the software development processes.

El-Emam et al. [28] performed a validation study to determine which OO design metrics were associated with fault-proneness. Their results indicate that an inheritance and an export coupling (EC) metrics were strongly associated with fault proneness.

Fioravanti and Nesi [34] have extracted over 200 different OO metrics to identify a suitable model for detecting fault-proneness of classes. They concluded that only a few of them were relevant for identifying fault prone classes.

Yu et al. [104] empirically evaluated a set of OO metrics in terms of their usefulness in predicting fault-proneness. Their validation is carried out using two data analysis techniques: regression analysis and discriminant analysis.

Subramanyam and Krishnan [93] validated the association between a subset of C&K metrics and faults detected during acceptance testing and those reported by customers. Based on industry data, results indicate that even after controlling the size of the software, they found that some of the measures in the CK suite of OO design complexity metrics significantly associated with faults.

2.1.2 Relationships between object-oriented design metrics and maintainability

Many researchers investigated the relationship between OO metrics and the maintainability of a software system. OO metrics was found correlated with software maintainability [5, 17, 33, 66, 72]. Thus, they can be used as good predictors of software maintainability.

Bandi et al. [5] empirically validated three existing OO design complexity metrics (Interaction Level, Interface Size, and Operation Argument Complexity), and assess their ability to predict maintenance time. Each of the three metrics by itself was found to be useful in predicting maintenance time.

Fioravanti and Nesi [33] presented metrics for prediction of adaptive maintenance effort, and validated them against real data by using statistical analysis. The validation has shown that several well-known metrics can be profitably employed for the prediction of maintenance effort. In another study, Misra [72] used a suite of twenty design/code measures to obtain indications of their effect on maintainability.

Li and Henry [66] validated the relationship of various metrics including all C&K metrics suite [22], except CBO, with maintenance effort in two commercial systems. They found a strong relationship between the metrics and maintenance effort in OO systems.

2.2 Class fault prediction

In OO systems, one approach to identify faulty classes early in development is to construct prediction models using OO design metrics. Predicting class fault proneness can be considered either a classification problem (e.g. a class is faulty or not), or a regression problem (e.g. fault density or number of faults).

Many research studies investigated the use of prediction models in fault prediction problem using OO design metrics. Most of these prediction models are built using statistical models [19, 28, 55, 104]. However, since the relationships between software metrics and quality factors are often complex and nonlinear, this limits the

accuracy of such models. As a result, research studies have investigated the use of computational intelligence models for software quality prediction due to their capabilities of modeling nonlinear functional relationships [29].

Research studies employed the use of computational intelligence models to predict class fault proneness, either as a regression or a classification problem. For regression problems, artificial neural networks (ANNs) have been used [44, 95] as well as decision trees [44]. For classification problems, ANNs have been also used [69].

Thwin and Quah [95] presented the application of neural networks in software quality estimation, in terms of predicting number of faults, using OO metrics. They used two neural network models, Ward neural network and general regression neural network (GRNN) to predict fault proneness classes using CK [22] and Tang et al. [94] metrics suite. Experimental results showed that OO metrics chosen in their study appear to be useful in predicting software quality. These software metrics are significantly related to the number of faults. In addition, GRNN network model is found to predict more accurately than Ward network model.

Gyimothy et al. [44] employed statistical (logical and linear regression) and machine learning (decision tree and neural network) methods to assess the applicability of the well-known OO metrics to predict the number of bugs in classes of Mozilla system. However, their results indicate that the precision of employed models is not yet satisfactory, and suggested combining multiple models (e.g. majority voting) as a future work.

Mahaweerawat et al. [69] proposed the use of ANNs as a new approach for predicting and classifying class faults in OO software systems. Multilayer perceptron neural network has been used to identify faulty classes, while radial basis function neural network is used to categorize the faults according to several defined fault types. It is concluded that the proposed model provides high accuracy in discrimination between faulty and fault-free classes.

It can be noticed that the number of investigated models for fault prediction at the class-level is limited, and these models are mainly statistical regression and neural networks.

2.3 Maintainability prediction

Research studies investigated the use of prediction models in software maintainability prediction. OO Software maintainability prediction models were constructed using OO metrics. Such models include TreeNet [30], multivariate adaptive regression splines [108], multivariate linear regression [108], multiple linear regression [62], naïve bayes [62], artificial neural network [95, 108], regression tree [62, 108], support vector regression [108].

Thwin and Quah [95] predicted the software maintainability as the number of lines changed per class. Their experimental results found that General Regression neural network predict more accurately than Ward network model.

Koten and Gray [62] evaluated and compared the naïve Bayes classifier with commonly used regression based models. The results suggest that the naïve bayes model can predict maintainability more accurately than the regression-based models **for one**

system, and almost as accurately as the best regression based model for the other system.

Zhou and Leung [108] explored the employment of multiple adaptive regression splines (MARS) in building software maintainability prediction models. MARS was evaluated and compared against multivariate linear regression models, artificial neural network models, regression tree models, and support vector models. The results suggest that for one system MARS can predict maintainability more accurately than the other four typical modeling techniques. Then, Elish [30] extended the work done by Zhou [108] to investigate the capability of TreeNet technique in software maintainability prediction. Their results indicate that TreeNet yielded improved, or at least competitive, prediction accuracy over previous maintainability prediction models.

2.4 Ensemble models

Both theoretical [46, 63] and empirical [48, 77, 78] research studies have demonstrated that a good ensemble is one where the individual prediction models in the ensemble are both accurate and make their errors on different parts of the input space.

Recently, ensemble models have received much attention and have demonstrated promising capabilities in improving classification accuracy over single classifiers [14, 92]. Some of these models are simple ensemble [92], AdaBoost [37], bagging [15] and boosting [38, 88]. Ensemble models have been used in the area of software engineering prediction problems. They have been used in software reliability prediction [107], software project effort estimation [14], and software module fault prediction [57]. In addition, they have been used in many real applications such as face recognition [43, 51],

OCR [70], seismic signal classification [90] and protein structural class prediction [10]. However, none of the ensemble models have been used in the area of class fault prediction and software maintainability prediction.

Neural networks ensembles received most attention from research community. Hansen and Salamon [46] showed that the generalization ability of a neural network system can be significantly improved through ensembling training several neural networks and combining their results. Sollich [92] defined neural network ensemble as a collection of a finite number of neural networks that are trained for the same task. Zhou et al. [109] proposed GASEN (Genetic Algorithm based Selective Ensemble) a neural network ensemble that utilizes all available neural networks to constitute an ensemble. They assigned random weights to neural networks and employs genetic algorithm GA to evolve the weights. Then, they select some neural networks based on the evolved weights to make up the ensemble. Empirical study shows that their ensemble is superior to both bagging and boosting in both regression and classification.

In summary, ensemble models demonstrate a high potential in providing reliable predictions. However, they have not been applied and evaluated in the context of identifying faulty classes in OO software, and predicting software maintainability.

2.5 Summary

This literature review presented the related work done in the area of predicting software quality as fault and maintainability prediction. We surveyed the computational intelligence models used in that area and we summarized these studies in the below table. In addition to the summary table, we came up with the following remarks:

- There are a limited number of studies done in the area of class fault prediction as a regression problem. Thus, more studies should be conducted toward this area.
- More computational intelligence models should be investigated to build more accurate software prediction models.
- OO metrics are found to be correlated with software quality, such as fault proneness and maintainability. Therefore, they can be used as early software quality indicators.
- Ensembles demonstrate a high potential in providing reliable predictions. Thus, they should be investigated in the context of software quality assessment

Study	Domain		Models used	Metrics	
				independent	dependent
Thwin and Quah [95]	Class Fault proneness	Regression	Artificial neural networks	C&K and Tang [94]metrics suite	Number of faults
Gyimothy et al. [44]		Regression	Artificial neural networks & Decision trees	C&K, one more OO metric (LCOMN), and traditional code-size metric (LOC)	Number of faults
Mahaweerawat et al. [69]		Classification	Artificial neural networks	C&K and Tang metrics suite	Binary (Faulty or not faulty)
Elish [30]	Maintainability	Regression	TreeNet	C&K, and Li & Henry [66] metrics	Maintenance effort
Zhou and Leung [108]		Regression	Multivariate adaptive regression splines (MARS), regression tree, and support vector machines	C&K, and Li and Henry metrics	Maintenance effort
Koten and Gray [62]		Regression	Naïve bayes	C&K, and Li and Henry metrics	Maintenance effort
Thwin and Quah [95]		Regression	Artificial neural networks	C&K, and Li and Henry metrics	Maintenance effort

Table 1: Literature review summary

CHAPTER 3

Technical background

This chapter gives an overview over the general technical background of the empirical studies conducted in this thesis. First, it gives the technical background of the investigated stand-alone prediction models, along with their parameters initialization. Next, it describes the datasets used across different experiments. After that, it formulates the evaluation measures used to evaluate the performance of prediction models. Finally, it sheds the light on different validation techniques exist in literature.

3.1 Individual prediction models

In this section, we describe the investigated individual prediction models that we used as base for our ensemble models. Selection of prediction models was based on a couple of criteria. We wanted to select models across different categories to achieve a balance between established prediction models, and we selected the models that are commonly and widely used in the literature of software quality prediction.

Different prediction models may be grouped into different categories of statistical approaches, neural networks, support vector machines, and tree based models. The following table presents the selected individual prediction models:

Prediction model	abbreviation	Category	Type
Multilayer Perceptron	MLP	Neural network	Classification and regression
Radial Basis Function Network	RBF	Neural network	Classification and regression
Bayesian Belief Network	BBN	Statistical	Classification
Naïve Bayes	NB	Statistical	Classification
Support Vector Machines	SVM	Support vector machine	Classification
Support Vector regression	SVMreg	Support vector machine	Regression
Decision Tree	DT	Tree	Classification
M5 Model tree	M5P	Tree	Regression

Table 2: Selected individual prediction models

3.1.1 Artificial Neural Networks

Artificial Neural Network (ANN) [49, 53] is a mathematical model or computational model consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. ANN is a non-linear statistical data modeling tools used to model complex relationships between inputs and outputs. ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

ANN architecture consists of several layers of computing nodes; input layer, hidden layers and output layer. ANN can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs.

Artificial neurons, shown below, take a set of inputs x_1, \dots, x_n , along with their weights, to produce the output O using the following formula:

$$O = f(\text{network}) = f(\sum_{j=1}^n w_j x_j) \quad (1)$$

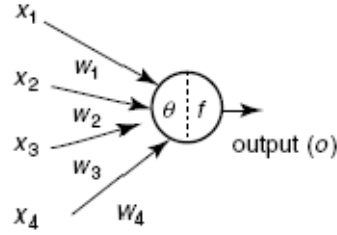


Figure 3: Artificial neuron [2]

Where w_j is the weight vector, and the function $f(\text{network})$ is referred to as an activation function. The variable network is defined as a scalar product of the weight and input vectors,

$$\text{network} = w_1 x_1 + \dots + w_n x_n \quad (2)$$

Based on various ANN's architectures, ANN can be classified into two categories:

- Feed-forward networks, in which graphs have no loops in network
- Recurrent (feedback) networks, in which we have loops in network

The most famous networks in feed-forward networks are multilayer perceptron (MLP), and radial basis function (RBF) networks. Their neurons are organized into layers that have unidirectional connections between them as shown in the figure below:

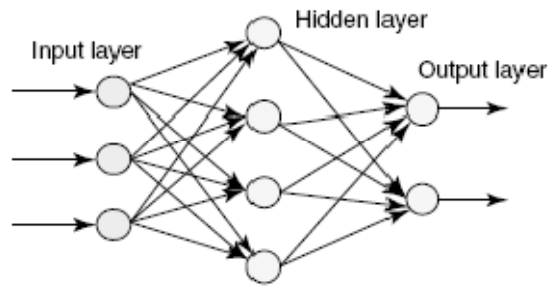


Figure 4: Feed-forward artificial neural network architecture [2]

3.1.1.1 Multilayer Perceptron (MLP)

Multilayer Perceptron [49, 53], shown below, is a feed-forward network that consists of an input layer, one or more hidden layers of nonlinearly activating nodes and an output layer. Each node in one layer connects with a certain weight to every other node in the following layer, and no computation is involved in the input layer.

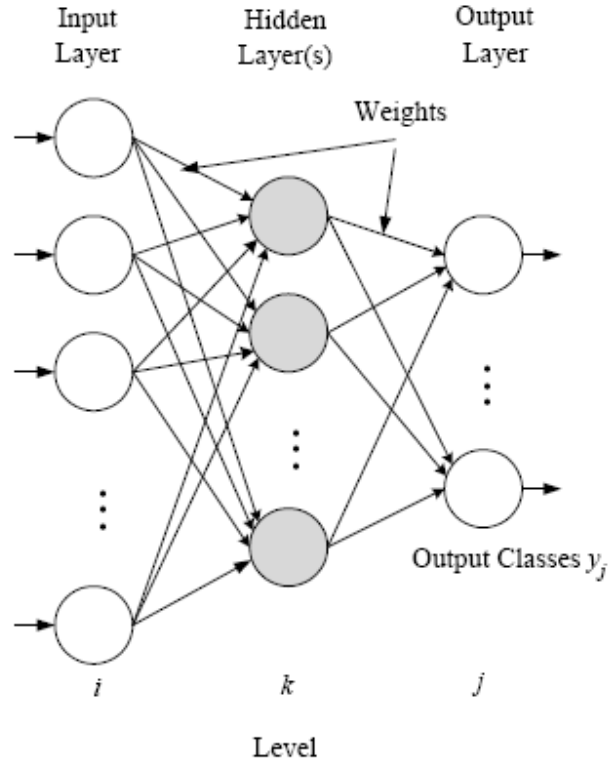


Figure 5: Example of a multilayer perceptron network [102]

MLP neuron, shown below, computes the weighted sum of the inputs at the presence of the bias, and passes this sum through the activation function. This process is described as follows:

$$y_j = f_j(v_j) \quad (3)$$

$$v_j = \sum_{i=1}^p w_{ji}x_i + \theta_j \quad (4)$$

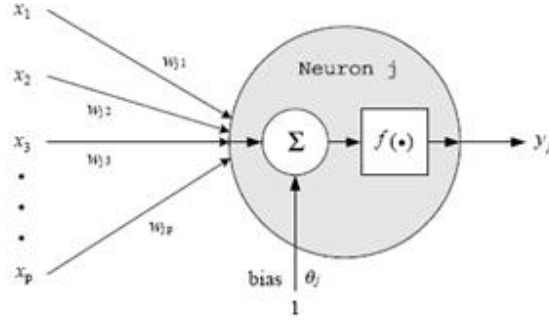


Figure 6: MLP neuron [102]

where \mathbf{v}_j is the linear combination of inputs $\mathbf{x}_1, \dots, \mathbf{x}_p$, θ_j is the bias, w_{ji} is the weight between the input \mathbf{x}_i and the neuron j , and $f_j(\cdot)$ is the activation function of the j th neuron, and y_j is the output.

The sigmoid function is a common choice of the activation function. The bias term θ_j contributes to the left or right shift of the sigmoid activation function, depending on whether θ_j takes a positive or negative value. The sigmoid function is calculated using this formula:

$$f(a) = \frac{1}{1+e^{-a}} \quad (5)$$

MLP uses backpropagation algorithm as the standard learning algorithm for any supervised-learning. Backpropagation algorithm requires that all activation functions used by the artificial neurons must be differentiable. It is used to calculate the error gradient of the network with respect to its modifiable weights, and find weights that minimize the error [45].

In backpropagation algorithm, choosing the learning rate is difficult, if the learning rate is set small enough to minimize the total error, the learning process will be slowed down. On the other hand, a larger learning rate may speed up learning process at the risk of potential oscillation. However, backpropagation algorithm uses momentum term to avoid oscillation problems during the search for the minimum error value [4].

3.1.1.2 Radial Basis Function Network (RBF)

Radial Basis Function network [21, 79, 80] is a three layer feed-forward network, as shown below, that uses a linear activation function for the output units and a nonlinear activation function for the hidden layer neurons. Input to each hidden neuron is the distance between the network inputs and center of that neuron's activation function. The simplest way to define centers setting it randomly to the training inputs, but this approach is prone to over fitting. An alternative is to cluster the training patterns into groups according to some similarity measurement and then assigning nodes to each cluster. The typical method to determine such clusters is by the k means clustering algorithm.

Figure 7 shows the general architecture of an RBF network with k input vector x, then the network computes the output as a scalar value using the following formula:

$$Y = f(\text{network}) = w_0 + \sum_{i=1}^m w_i \varphi(D_i) \quad (6)$$

Where w_0 is the bias, w_i is the weight parameter, n is the number of nodes in the hidden layers of the RBF neural network, and $\varphi(D_i)$ is the radial basis function.

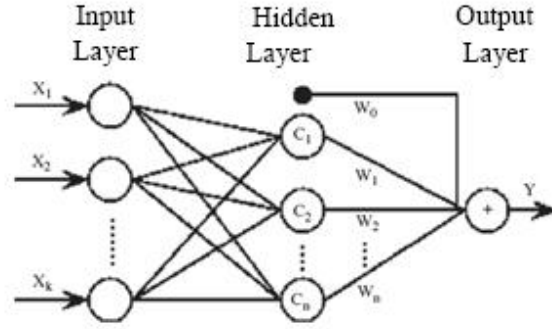


Figure 7: Radial basis function network [103]

RBF uses radial basis functions as activation functions to provide a flexible way to generalize linear regression function. Commonly used types of radial basis functions include Gaussian, Multiquadric, and Polyharmonic spline. However, RBF models with Gaussian basis functions intend to possess desirable mathematical properties of universal approximation and best approximation, and it is calculated as follows:

$$\varphi(D_i) = \exp(-D_i/\sigma_i) \quad (7)$$

Where σ is the radius of the cluster represented by the center node, the D_i represents the distance between the input vector X and all the data centers. Usually, the Euclidean norm is used to calculate distance as follows:

$$D_i = \sqrt{\sum_{j=1}^k (x_j - c_{ji})^2} \quad (8)$$

Where c is a cluster center for any of the given nodes in the hidden layer.

3.1.2 Statistical

Statistical classifiers strive to construct a Bayes optimal classifier by estimating either posterior probabilities directly, or class conditional probabilities. Both Bayesian belief

network and naïve bayes estimated class conditional probabilities, which are subsequently converted into posterior probabilities using Bayes theorem. These bayes networks are only applicable in the classification problem domain.

3.1.2.1 Bayesian Belief Network (BBN)

Bayesian Belief Network [40, 54] is a probabilistic directed acyclic graph that represents a set of random variables and their probabilistic independencies. The variable values can be discrete or continuous values according to a probability distribution, which can be different for each variable. In BBN graph, each node represents a random variable, while the directed edges between the nodes represent probabilistic dependencies among the corresponding random variables.

Figure 8 shows an example of a Bayesian belief network consisting of three variables, X_1 , X_2 , and Y . In the below figure, a variable is shown as an ellipse and a directed edge is shown as an arrow. This example shows that X_1 and X_2 has an association or a causal relationship with Y , (i.e. outcomes of the events X_2 and X_2 have an effect on the outcome of the event Y).

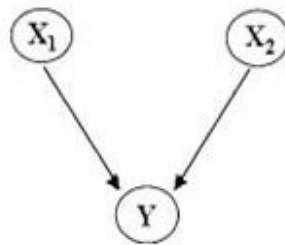


Figure 8: Example of Bayesian Belief Network

In a Bayesian belief network, a relationship between events is defined as a conditional probability, $P(Y/X)$, which is the probability of the variable Y conditional on a given outcome of variable X . The conditional probability is calculated using Bayes' Theorem [54]:

$$P(Y/X) = \frac{P(X/Y) P(Y)}{P(X)} \quad (9)$$

Where $P(X/Y)$ is the conditional probability of the variable X given the variable Y , and $P(X)$ and $P(Y)$ are the probabilities of variables X and Y , respectively. This probabilistic dependency is maintained by the conditional probability table (CPT), which is assigned to the corresponding variable.

3.1.2.2 Naïve Bayes (NB)

Naïve Bayes [42, 64, 86] is the simplest form of Bayesian network, in which all attributes are independent given the value of the class variable (i.e. conditional independence). The naïve bayes structure is unique, since it has one root node (called parent), representing the class node, and several independent children, corresponding to attribute nodes. Therefore, in presence of a training set we should only compute conditional probabilities in a frequent manner.

Once the Naïve Bayes is constructed, it can be used to classify any new instance characterized by a set of attributes $\mathbf{x}_1, \dots, \mathbf{x}_n$ by computing for each possible class c_i its posterior probability and then by taking the highest one. More formally the most probable class C is computed as follows:

$$C = \arg_{c_i} \max p(c_i) \prod_{j=1}^n p(x_j/c_i) \quad (10)$$

Although independence is generally a poor assumption, in practice naïve bayes competes well with more sophisticated classifiers, especially where the features are not strongly correlated [64].

3.1.3 Support Vector Machines

Support Vector Machines (SVMs) [24, 25, 96], proposed by Vapnik et al. [96], are a group of supervised learning methods, whose idea is based on the structured risk minimization (SRM) principle. Recently, it gained wide popularity due to its many attractive features and promising empirical performance. The main advantage of SVM is that it adopts the structure risk minimization (SRM) principle, which has been shown to be superior to the traditional empirical risk minimization (ERM) principle, employed by conventional neural networks [96].

Support vector machine (SVM) was originally developed for solving the classification problems [24, 96], but recently it was extended to the domain of regression problems [91, 97].

3.1.3.1 Support Vector Machines (SVM)

SVMs main aim is to minimize the empirical error and maximize the geometric margin. SVMs try to separate a given set of binary labeled training data with a hyperplane that is maximally distant from them (i.e. maximum margin hyperplane). As Figure 9 shows, the input space of N training data points $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)$ can be separated by a hyperplane $\mathbf{H} : \mathbf{w}^T \mathbf{x} + \mathbf{b} = 0$. This hyperplane H is located by determining two parallel hyperplanes H_1, H_2 that have the maximum margin $\frac{2}{\|\mathbf{w}\|^2}$ with the conditions that there are no data points between them.

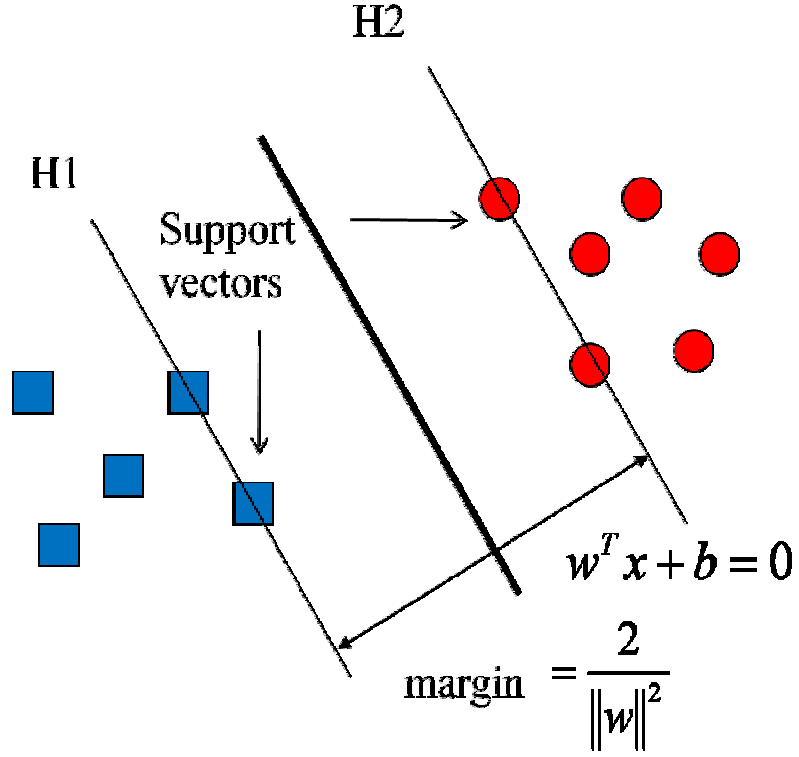


Figure 9: Maximal margin hyperplane

In case of training data is linearly separable by hyperplane, we separate the data with the maximal margin hyperplane as follows:

$$\text{minimize}_w \quad \frac{1}{2} w^T w \quad (11)$$

$$\text{subject to: } y_i(w^T x_i + b) \geq 1, i = 1, \dots, L$$

where (x_i, y_i) is the training data, and L is the total number of training sets.

In case of training data is linearly non-separable, we want to separate the training data with a minimal number of errors. This yields the introduction of positive slack variables $\delta_i \geq 0$ in the constraints to measure how much the margin constraints are violated:

$$\text{minimize}_{w, \delta} \quad \frac{1}{2} w^T w + C \sum_{i=1}^L \delta_i \quad (12)$$

$$\text{subject to:} \quad \delta_i \geq 0$$

$$y_i(w^T x_i + b) \geq 1 - \delta, i = 1, \dots, L$$

where C is the regularizing (margin) parameter that determines the trade-off between the maximization of the margin and minimization of the classification error.

Various kernel functions are employed in order to transform the data into a non-linear feature space, as shown in the below figure. The hyperplane found by the SVM training algorithm in the transformed feature space corresponds to a non-linear decision boundary in the initial input space. The most famous kernel functions are radial basis function (RBF) and polynomial.

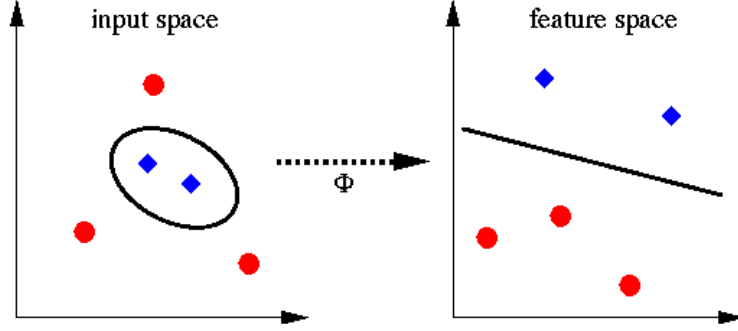


Figure 10: Data transformation into a non-linear space

3.1.3.2 Support vector regression (SVMreg)

The basic idea of SVM for regression (SVMreg) is to introduce kernel function, map the input space into a high-dimensional feature space via a nonlinear mapping and to perform a linear regression in this feature space [96].

Suppose we are given a set of L training data $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i th input pattern from the d dimension input space and has a corresponding target value $y_i \in \mathbb{R}$ for $i = 1, \dots, L$, where \mathbb{R} is the set of real number. The goal of support vector regression (SVMreg) is to find a function that approximates the actually obtained targets y_i for all the training data, and has a minimum generalization error. The general form of a SVMreg function can be given by:

$$f(x) = w * \phi(x) + b \quad (13)$$

where $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, $*$ denotes the dot product in \mathbb{R}^n , and ϕ is a non-linear transformation from \mathbb{R}^d to the high dimensional space \mathbb{R}^n . Our goal is to determine the value of w and b such that $f(x)$ can be determined by minimizing the regression risk

$$R_{reg}(f) = C \sum_{i=0}^L \delta(f(x_i) - y_i) + \frac{1}{2} \|w\|^2 \quad (14)$$

where δ is a cost function, C is a constant that represents penalties for estimation error. A heavier penalty trains the regression to minimize errors by making fewer generalizations. SVMreg function can be reformulated as:

$$f(x) = \sum_{i=1}^L (\alpha_i - \alpha_i^*) k(x_i, x) + b \quad (15)$$

where $k(x_i, x)$ is called the kernel function. Common kernel functions include Gaussian, linear, and polynomial.

When applying SVMreg in real applications, we need to give a kernel function and the penalty C .

3.1.4 Tree

Tree algorithm models recursively partition the training data by means of attribute splits. These tree algorithms differ mainly in the splitting criterion, which determines the attributes to separate the data. We have chosen Decision Tree (DT) for classification problems, while M5 Model tree (M5P) for regression problems.

3.1.4.1 Decision Tree (DT)

Decision Tree [71, 84, 105] is a flow chart-like tree structure that does mapping from observations about an item to conclusions about its target value. Figure 11 show the general structure of a decision tree. In DT, the topmost node is the root node, each internal node denotes an attribute test, each branch represents an outcome of the test, and each leaf node represents classes.

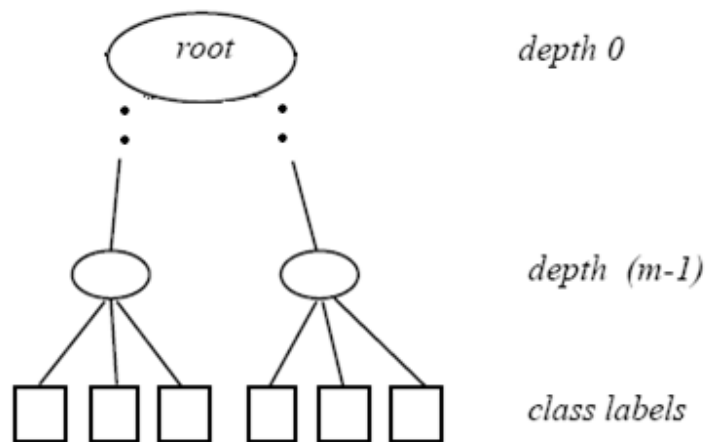


Figure 11: A decision tree [87]

Decisions trees are created typically using C4.5 algorithm developed by Quinlan [84]. C4.5 creates decision tree whose structure consists of leaves using a top-down, divide-and-conquer approach. C4.5 algorithm steps can be summarized in the following step:

- DT Construction: creates unpruned decision tree by recursively partitioning the data.
- Pruned decision tree: in this step the idea is to remove parts of the tree that do not contribute to classification accuracy, since the resulting unpruned decision tree is often complex and overfits the data.

3.1.4.2 M5 Model tree (M5P)

M5 Model tree [85, 100] is an algorithm for generating M5 model trees that predicts numeric values for a given instance. The algorithm requires the output attribute to be numeric while the input attributes can be either discrete or continuous.

To build a model tree as shown in Figure 12, using the M5 algorithm, we start with a set of training instances. The tree is built using a divide-and-conquer method. At a node, starting with the root node, the instance set that reaches it is either associated with a leaf or a test condition is chosen that splits the instances into subsets based on the test outcome. In M5 the test that maximizes the error reduction is used. Once the tree has been built, a linear model is constructed at each node. The linear model is a regression equation.

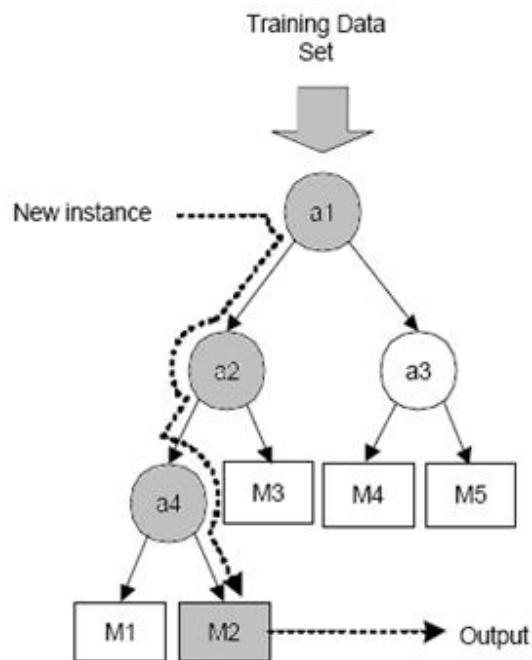


Figure 12: M5 Model tree [8]

As shown in Figure 12, where a_i are the split nodes and M_j are the models, if a new instance is given, the tree is traversed from top to bottom until a leaf node is reached. At each node in the tree a decision is made to follow a particular branch based on a test

condition on the attribute associated with that node. Each leaf has a linear regression model associated with the form:

$$w_0 + w_1 a_1 + \dots + w_k a_k \quad (16)$$

based on some of the input attributes $\mathbf{a}_1, \dots, \mathbf{a}_k$ in the instance and whose respective weights w_0, w_1, \dots, w_k are calculated using standard regression. As the leaf nodes contain a linear regression model to obtain the predicted output, the tree is called a model tree.

3.1.5 Parameter initialization

The parameters of the base prediction models were initialized with the default settings set by WEKA tool as follows:

- **MLP:** trained using backpropagation algorithm. Sigmoid was used as an activation function. Number of hidden layers was 5. Learning rate was 0.3 with momentum 0.2. Network was set to reset with a lower learning rate. Number of epochs to train through was 500. Validation threshold was 20.
- **RBF:** implements a normalized Gaussian radial basis function network. Random seed to pass on to K-means clustering algorithm was 1. Number of clusters for K-means clustering algorithm to generate was 2, with minimum standard deviation for clusters set to 0.1.
- **BBN:** used SimpleEstimator for estimating the conditional probability table of a Bayes network with 0.5 as the initial count for each value. Hill climbing algorithm was used as a search algorithm.
- **NB:** no parameters were required.

- **SVM:** cost parameter C was set to 1 and RBF was used as a kernel. The tolerance of termination criterion was 0.001.
- **SVMreg:** cost parameter C was set to 1, with polynomial as SVMreg kernel. WEKA implements various algorithms for parameter learning, it uses the most popular (RegSMOImproved) algorithm [89], as a default learner.
- **DT:** used C4.5 algorithm [84] to generate decision tree. Confidence factor used for pruning was set to 25%. Minimum number of instance per leaf was 2.
- **M5P:** used M5 algorithm for generating M5 model trees [85, 100]. Pruned M5 model trees were built, with 4 instances as the minimum number of instances allowed at a leaf node.

3.2 Datasets

Choosing the appropriate dataset in an empirical study is the most important and crucial step. We realized that choosing public datasets makes our research verifiable, repeatable, and reputable [26]. The following table summaries the characteristics of the public datasets used in this research:

Dataset name	Domain	Type	# observations	Language	Description
KC1	Fault proneness	Classification	145	C++	Storage management for receiving and processing ground data
KC1	Fault density	Regression	60	C++	Storage management for receiving and processing ground data
UIMS	Maintenance effort	Regression	39	Ada	User interface management system
QUES	Maintenance effort	Regression	71	Ada	Quality evaluation system

Table 3: Software quality datasets

3.2.1 Class fault datasets

KC1 dataset has two versions, that has the same independent variables, but different in dependent variables. The first dataset is for class fault proneness, while the second dataset is for class fault density estimation.

KC1 (classification) is a class level defect dataset from a mission critical NASA software project, which is publicly available from the repository of the NASA IV&V Facility Metrics Data Program [1]. This project is a storage management system for receiving and processing ground data. It is written in C++ and consists of 145 classes with more than 43 KLOC. This dataset consists of class-level metrics data and the associated fault data that has been collected since the beginning of the project.

The dependent (output) variable “FAULTY” is a binary variable indicating whether or not a class is faulty. Around 40% of the classes are faulty. The independent (input) variables are nine class-level metrics. Six out of these nine metrics are the well-known Chidamber and Kemerer’s metrics [22], i.e., Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling Between Object classes (CBO), Response For a Class (RFC), and Lack of Cohesion in Methods (LCOM). The other three metrics are Fan-in (FIN), Percentage of Public/Protected Data (PPD), and Dependency on Child (DOC). Table 4 provides brief description for each metric.

The other version of KC1 is used for class fault density estimation. KC1 (regression) share the same independent variables as in KC1 (classification), but they differ in dependent variable and number of observations. Dependent (output) variable for KC1 (regression) is the class fault density, which is measured as the total number of faults divided by total size (LOC), and the total number of observations are 60.

Metric	Description
WMC	Count of methods implemented within a class
DIT	Level for a class within its class hierarchy
NOC	Number of immediate subclasses of a class
CBO	Number of distinct non-inheritance-related classes on which a class depends
RFC	Count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance
LCOM	The average percentage of methods in a class using each data field in the class subtracted from 100%
FIN	Number of classes that depend upon a class
PPD	Percentage of public and protected data in a class
DOC	Whether a class is dependent on a descendant
Faulty	Binary variable: a class is faulty, or not faulty
Fault Density	The number of faults divided by total size

Table 4: KC1 independent and dependent variables

3.2.2 Maintainability datasets

We used two popular OO maintainability datasets published by Li and Henry [66]: UIMS and QUES datasets. The UIMS dataset contains class-level metrics data collected from 39 classes of a user interface management system, whereas the QUES dataset contains the same metrics collected from 71 classes of a quality evaluation system. Both systems were implemented in Ada. Both datasets consist of eleven class-level metrics: ten independent variables and one dependent variable.

The independent (input) variables are five Chidambar and Kemerer metrics [22]: WMC, DIT, NOC, RFC, and LCOM; four Li and Henry metrics [66], i.e., Message Passing Coupling (MPC), Data Abstraction Coupling (DAC), Number Of Methods (NOM), and Number of properties (SIZE2); and one traditional lines of code metric (SIZE1). Table 5 provides brief description for each metric.

The dependent (output) variable is a maintenance effort represented by measure “CHANGE”, which is the number of lines in the code that were changed per class during a 3-year maintenance period. A line change could be an addition or a deletion. A change in the content of a line is counted as a deletion and an addition.

The analysis done in [30] on both datasets, indicates that both datasets have different characteristics, and therefore, considered heterogeneous and a separate maintainability prediction model is built for each dataset.

Metric	Description
WMC	Count of methods implemented within a class
DIT	Level for a class within its class hierarchy
NOC	Number of immediate subclasses of a class
RFC	Count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance
LCOM	The average percentage of methods in a class using each data field in the class subtracted from 100%
MPC	The number of messages sent out from a class
DAC	The number of instances of another class declared within a class
NOM	The number of methods in a class
SIZE1	The number of lines of code excluding comments
SIZE2	The total count of the number of data attributes and the number of local methods in a class
Change	The number of lines added and deleted in a class, change of the content is counted as 2

Table 5: UIMS & QUES independent and dependent variables

3.3 Tool

WEKA, short for Waikato Environment for Knowledge Analysis), is a software tool used to conduct this research [101]. It is an open source tool, implemented in java, that contains a collection of machine learning algorithms for data mining tasks and tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

The use of an open source tools should be encouraged, since it makes the findings more open to the public, and increases the contribution to the research.

3.4 Evaluation measures

In this section, we will describe the evaluation measures used in evaluating prediction models performance. Evaluation measures can be divided into two categories (i.e. classification and regression). The selection of these measures varied from one study to another. We selected the measures based on the most commonly and widely used in the literature. Future research should use the most commonly used measures to make their findings more consistent with others.

3.4.1 Classification evaluation measures

The evaluation measures of a prediction model for binary classification problems (i.e., correctly or not correctly classified cases) are derived from a confusion matrix, like the one shown in Table 6. The confusion matrix has four categories [27]: True positives (TP) are cases correctly classified as positives. False negatives (FN) are positive cases incorrectly classified as negative. False positives (FP) are negative cases incorrectly classified as positive. Finally, true negatives (TN) refer to negative cases correctly classified as negative.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 6: Confusion matrix for binary classification problems

3.4.1.1 Accuracy

Accuracy, also known as correct classification rate, can be seen as proportion of true results (i.e. both true positives and true negatives) in the population. It is calculated as the sum of correct classifications divided by the total number of classifications [101]:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (17)$$

3.4.1.2 Recall

Recall, also known as true positive rate and sensitivity, is the percentage of true positives that are classified correctly. It is calculated as follows [101]:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (18)$$

3.4.1.3 Precision

Precision, also known as specificity, is the ratio between the number of correctly identified positives and predicted positives. It is calculated as follows [101]:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (19)$$

3.4.1.4 F-measure

F-measure is the harmonic average of precision and recall. F-measure is commonly used measure along with precision and recall, since it integrates the trade-off between precision and recall [101]. It is calculated as follows:

$$\text{F_measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (20)$$

In all previous measures, the higher the value is, the better prediction model will be.

3.4.1.5 ROC

Receiver operating characteristics (ROC) graphs are useful for organizing classifiers and visualizing their performance. ROC graphs are another way besides confusion matrices to examine the performance of classifiers, they have been increasingly used in computational intelligence research [31].

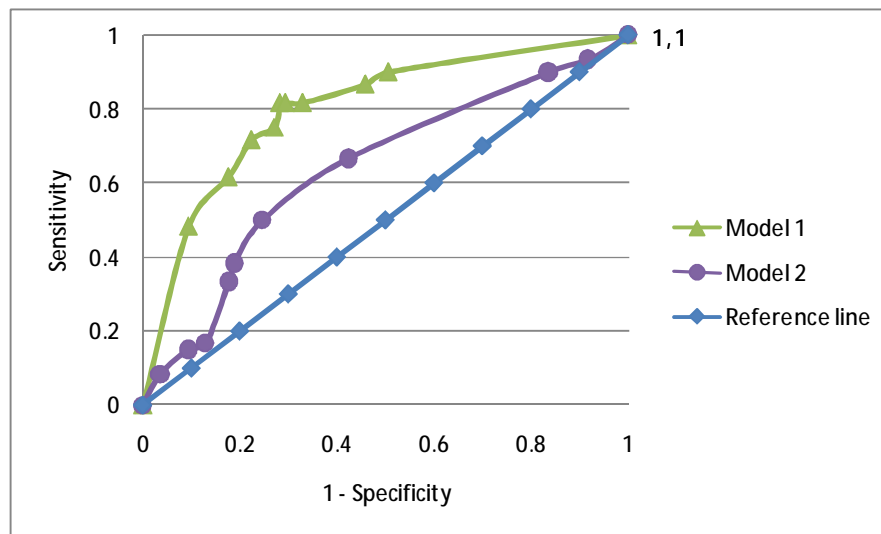


Figure 13: A ROC graph

ROC graphs, shown for two prediction models A and B in Figure 13, plot the false positive rate (x-axis) (i.e. 1- specificity), versus true positive rate (y-axis) (i.e. sensitivity). Each point represents the output of the prediction model, with respect to the threshold. ROC graphs description is summarized in the following points:

- Point (0,1) corresponds to perfect classification. Better models intend to have (FP, TP) values closer to this point.
- Point (1,0) is a classifier which misclassifies every case
- Point (1,1) is a classifier that classifies every case as positive
- Point (0,0) is a classifier which classifies every case as negative

However, in ROC graph, if two models curve intersects, we rely on Area Under Curve (AUC) measure to compare their performance. Model with higher AUC, is expected to perform better. The reference line in Figure 13 represents model with (0.5) AUC.

We will list all equations required to plot the ROC curve, and calculating the AUC [13]:

$$\text{False positive rate} = 1 - \text{specificity} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{True positive rate} = \text{sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$AUC = \sum_i \left\{ (1 - \beta_i * \Delta\alpha) + \frac{1}{2} [\Delta(1 - \beta) * \Delta\alpha] \right\} \quad (21)$$

where:

$$\alpha = \text{False positive rate},$$

$$1 - \beta = \text{True positive rate}$$

$$\Delta(1 - \beta) = (1 - \beta_i) - (1 - \beta_{i-1})$$

$$\Delta\alpha = \alpha_i - \alpha_{i-1}$$

3.4.2 Regression evaluation measures

Many regression measures have been proposed to evaluate regression prediction models. However, most of the commonly used regression measures are derived from magnitude of relative error (MRE). MRE was introduced in 1986 by Conte et al.[23]. Since then, it

became the standard evaluation criterion to assess software prediction models. It is defined as follows:

$$\mathbf{MRE}_i = \frac{|Actual_i - Predicted_i|}{Actual_i} \quad (22)$$

The regression measures that are based on MRE are: mean magnitude of relative error (MMRE), Standard deviation magnitude of relative error (StdMRE), and prediction at level q (Pred(q)) measures. Since StdMRE, MMRE and Pred(k) are well-known evaluation criteria, they have been adopted as evaluation measures for regression models.

3.4.2.1 MMRE

MMRE calculates the MRE value for each observation i predicted. The aggregation of MRE over multiple observations (N) can be achieved through the Mean MRE (MMRE) as follows [59]:

$$\mathbf{MMRE} = \frac{1}{N} \sum_i^N MRE_i \quad (23)$$

3.4.2.2 StdMRE

Standard deviation magnitude of relative error is less sensitive to the extreme values compared to the MMRE, and it is more likely to select the true model based on StdMRE if the underestimate is served [35].

3.4.2.3 Pred(0.3)

Pred(q) is the percentage of observations whose MRE is less than or equal to level q. It is calculated as follows [32]:

$$\mathbf{Pred(q)} = \frac{k}{n} \quad (24)$$

Where k is the number of observations whose MRE is less than or equal to the selected level q , and n is the total number of observations in the dataset. An acceptable value for level q is 0.3, as indicated in the literature [23, 62, 108]. Therefore, we adopted this value in our empirical studies. Furthermore, for a regression model to be considered accurate, it is suggested in the literature that $\text{Pred}(0.3)$ value should be larger than or equal to 70% [32, 68].

3.5 Validation techniques

In this section, we will give an overview of the most used validation techniques in the literature. Validation techniques are used to evaluate the performance of prediction models. These techniques can be categorized into two categories: cross validation and holdout method.

3.5.1 Cross validation

In k -fold cross validation, we resample train and test set k times. This is explained in the following steps:

- Randomly divide dataset into k equal-sized folds
- Train on $k-1$ folds, test on remaining fold
- Repeat to use each fold once for testing

3.5.1.1 10 fold cross validation

A 10 fold cross validation [60] (i.e. k -fold cross validation, with k set to 10), is a common validation technique used to evaluate the performance of the prediction models. In 10 fold cross validation; a dataset is randomly partitioned into 10 folds of equal size. For 10 times, 9 folds are picked to train the models and the remaining fold is used to test them, each time leaving out a different fold.

3.5.1.2 Leave-one-out

A leave-one-out cross-validation (LOOCV) procedure is another commonly used validation technique; it is a special type of k folds cross validation, whereas k is set to the total number of dataset observations. In this procedure, one observation is removed from the dataset, and then each model is built with the remaining $n-1$ observations and evaluated in predicting the value of the observation that was removed. The process is repeated each time removing a different observation. This procedure has a number of advantages [74, 101]: (i) it is closer to a real world situation than k -cross validation from a practitioner's point of view (ii) it is deterministic (no sampling is involved); and (iii) it ensures the largest possible amount of data is used for training.

3.5.2 Holdout method

Holdout method splits the dataset into two sets, called the training set and the testing set. The most common split percentage is 70% for training and 30% for testing [74].

CHAPTER 4

Single model ensembles (Classification)

4.1 Single model ensembles for classification

In classification, a single model ensemble (i.e. single classifier ensemble), as shown below, consists of a set of individually trained classifiers of the same type whose classifications are combined to produce the final classification. The simplest way to combine classifiers is by voting. Bagging [15] and boosting [36] are well known single model ensemble models, which implement this approach in different ways using a single base classifier.

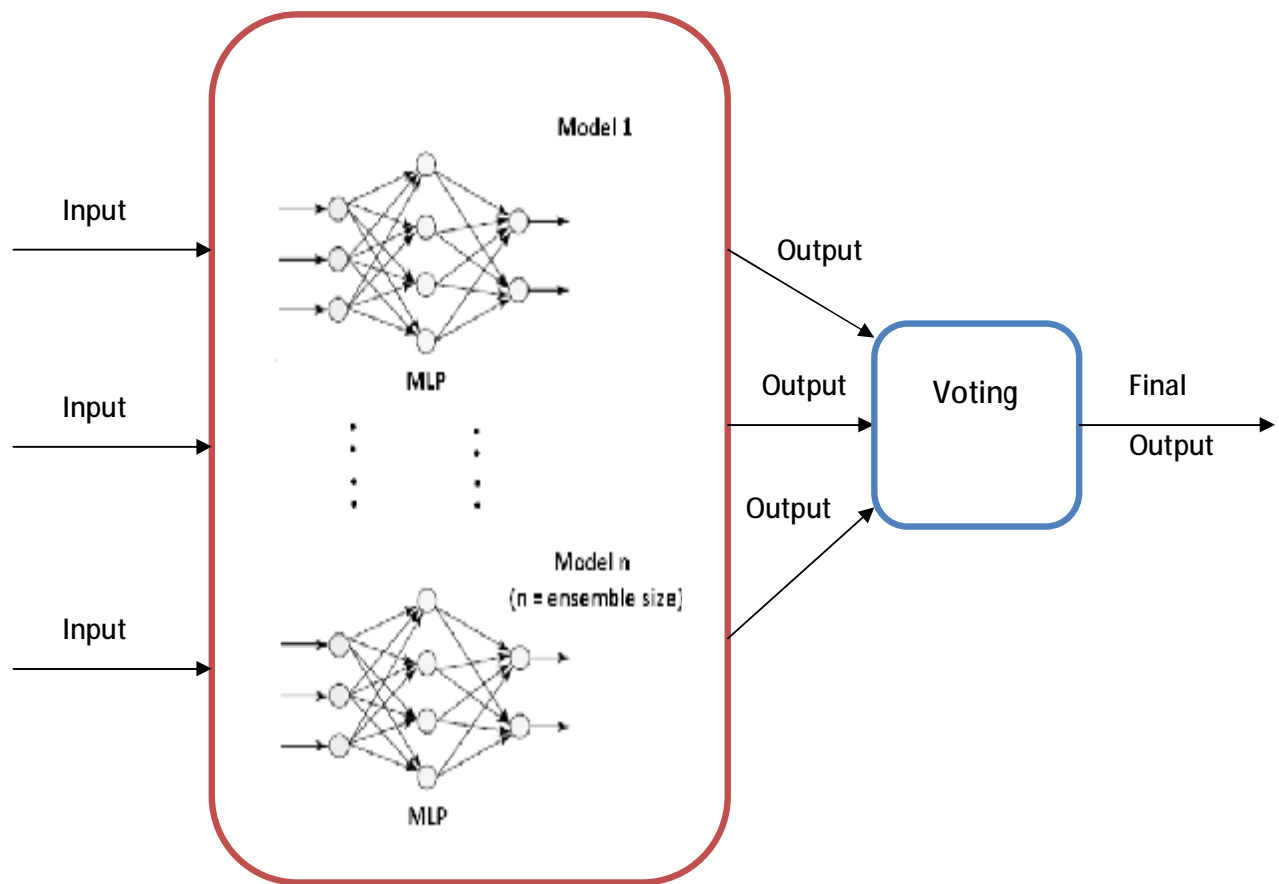


Figure 14: Single classifier ensemble

4.1.1 Bagging

Bagging, short for bootstrap aggregating, is an ensemble technique proposed by Breiman [15] to improve the accuracy of classification models by combining classifications of same type (i.e., based on the same base classifier) of randomly generated training sets. Bagging assigns equal weight to models created, thus helps in reducing the variance associated with classification, which in turn improves the classification process. Bagging technique has presented good results whenever the learning algorithm is unstable [15]. The following figure states the bagging algorithm [101]:

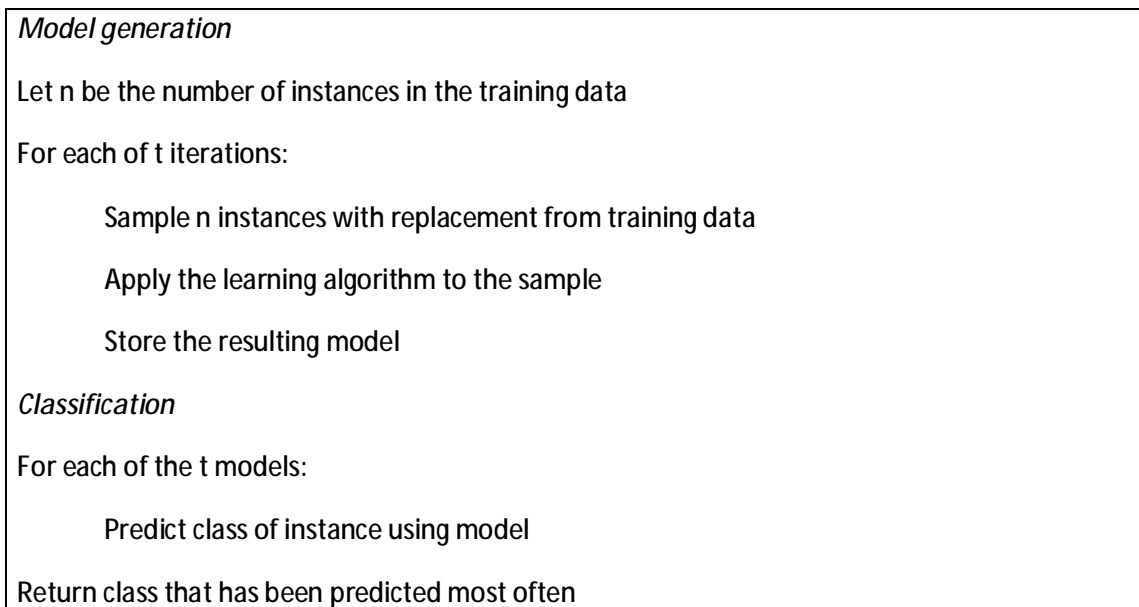


Figure 15: Bagging algorithm

Bagging technique requires three parameters:

- classifier: the base classifier to apply bagging on.
- bagSizePercent: size of each bag, as a percentage of the training set size.
- numIterations: number of instances of the base classifiers to be created, i.e. the

ensemble size. In this thesis, we prefer to use the term *ensemble size* for clarity.

4.1.2 Boosting

Boosting is an ensemble technique proposed by Freund [36] to build a classifier ensemble incrementally, by adding one classifier at a time. The training set used for each member of the ensemble is chosen based on the performance of the earlier classifiers in the ensemble. The following figure states the boosting algorithm [101]:

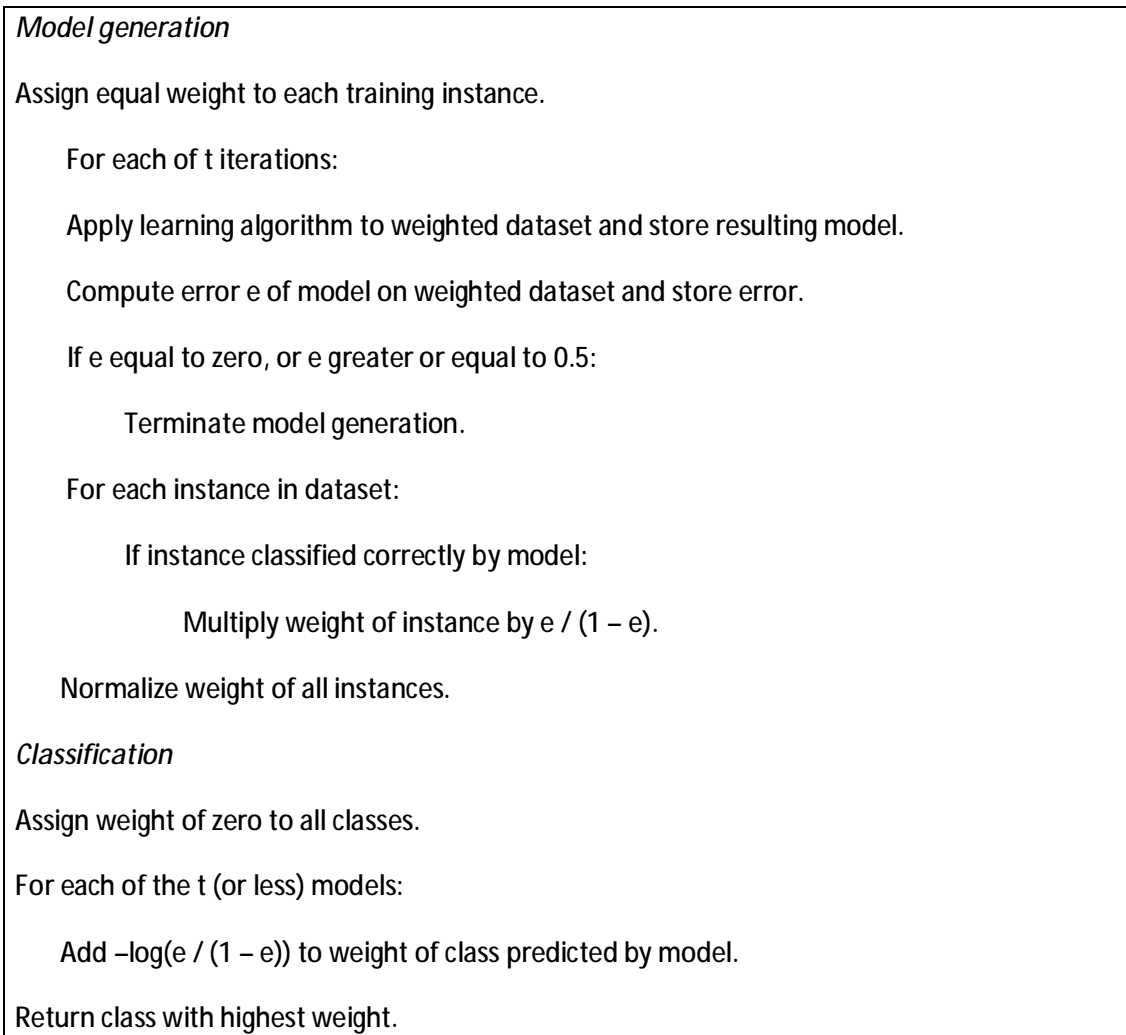


Figure 16: Boosting algorithm

Boosting technique requires three parameters:

- classifier: the base classifier to apply boosting on.
- Resampling/Reweighting: which approach is used (resampling or reweighting)
- numIterations: number of instances of the base classifiers to be created, i.e. the *ensemble size*. In this thesis, we prefer to use the term ensemble size for clarity purpose.

There are a family of boosting algorithms [39]. In this thesis, we used AdaBoost algorithm proposed by Freund et al. [37]. AdaBoost was proposed to improve other learning algorithms performance. There are two approaches implemented in AdaBoost: resampling and reweighting. In resampling, the fixed training sample size and training examples are resampled according to a probability distribution used in each iteration. In reweighting, all training examples with weights assigned to each example are used in each iteration to train the base classifier. In this paper, we used the resampling approach, because it has been reported to yield better accuracy [6, 106].

There is no conclusion on which ensemble technique is superior to the other. However, several observations have been made [6, 10, 76, 83]:

- In some cases, boosting can significantly outperform bagging, while in some other cases, it can also be substantially worse than bagging (in a few cases even worse than individual classifiers)
- bagging's improvement over individual classifiers is more consistent on various data sets than boosting's.

The following table summarizes the similarities and differences between bagging and boosting techniques:

	Bagging	Boosting
Similarities	<ul style="list-style-type: none"> • Uses voting • Combines models of the same type 	
Differences	Individual models are built separately	Each new model is influenced by the performance of those built previously
	Equal weight is given to all models	Weights a model's contribution by its performance

Table 7: Comparison between bagging and boosting

4.2 Experiment design

4.2.1 Goal

The main goal of this experiment is to determine the extent to which bagging and boosting ensemble techniques offer an increase in classification accuracy over single classifiers in the context of identifying faulty classes in OO software.

4.2.2 Settings

The following table presents the settings for the conducted experiment:

Experiment type	Classification
Investigated models	BBN
	NB
	MLP
	RBF
	SVM
	DT
Data set	KC1
Evaluation measure	Accuracy
Validation technique	Leave-one-out (LOO)

Table 8: Experiment settings for single classifier ensemble (classification)

4.2.3 Bagging and boosting parameters

Parameters of the bagging and boosting ensemble techniques were initialized as follows. In bagging, the bag size was set to 100. In boosting (Adaboost), resampling was used instead of reweighting. Boosting by resampling improves the classification accuracy, the execution speed as well as the robustness to classification noise [106]. We have applied bagging and boosting to all of the investigated classification models. We used ensemble sizes from 5 to 50 with an increment of 5. This approach was used previously in the literature [15, 99]. Hence, we can investigate the effect of the ensemble size on the classification accuracy of these ensemble techniques.

4.3 Results

In this section, we present and analyze the results obtained from the conducted experiment. First, we compare the classification accuracy of the investigated classification models as base classifiers without applying ensemble techniques on them. Then, we compare the classification accuracy results after applying bagging and boosting techniques on each base classifier. Finally, we discuss the effect of applying bagging and boosting on each base classifier.

Table 9 shows the classification accuracy for each of the investigated classification models without applying ensemble techniques on them, i.e., classification accuracy of single classifiers. It is clear that MLP outperformed all other models in accuracy. However, SVM and BBN produced the worst accuracy. There is a significant difference in accuracy, nearly 9%, between the highest accuracy (78.62%) achieved by MLP and the second highest accuracy (69.66%) achieved by NB.

Model	Accuracy
MLP	78.62
RBF	68.28
BBN	62.76
NB	69.66
SVM	62.76
DT	63.45

Table 9: Classification accuracy of single classifiers

Table 10 presents the achieved accuracy by each classification model after applying bagging on it using different ensemble size. Bold values indicate that bagging resulted in improved accuracy over the single classifier reported in Table 4. It can be observed that bagging improved the accuracy of MLP, RBF, BBN, NB and DT. However, bagging SVM resulted in lower accuracy compared to single SVM in general. In case of RBF and DT, it can be observed that bagging increased their classification accuracy regardless of the ensemble size. However, in case of MLP and BBN, bagging increased their classification accuracy when the ensemble size is 20 or more.

Table 11 presents the achieved accuracy by each classification model after applying boosting on it using different ensemble size. Bold values indicate that boosting resulted in improved accuracy over the single classifier reported in Table 4. Clearly, boosting improved the accuracy of RBF, BBN and DT regardless of the ensemble size. However, applying boosting on MLP, NB and SVM reduced their classification accuracy regardless of the ensemble size.

Ensemble size	MLP	RBF	BBN	NB	SVM	DT
5	74.48	69.66	57.24	69.66	62.07	66.90
10	77.93	72.41	62.07	70.34	62.07	67.59
15	78.62	71.03	61.38	68.97	60.69	68.28
20	79.31	73.10	63.45	70.34	61.38	68.97
25	80.00	71.72	64.83	70.34	62.07	68.97
30	80.00	72.41	65.52	70.34	62.76	69.66
35	80.00	72.41	64.83	70.34	62.76	70.34
40	80.00	73.10	66.21	70.34	62.76	69.66
45	80.00	72.41	67.59	70.34	62.07	71.72
50	80.00	73.10	66.90	70.34	62.07	71.72

Table 10: Classification accuracy after applying bagging on base classifiers using different ensemble size

Ensemble size	MLP	RBF	BBN	NB	SVM	DT
5	73.79	72.41	64.83	67.59	60.00	70.34
10	74.48	75.17	64.83	68.28	60.69	70.34
15	73.79	74.48	64.83	69.66	62.07	73.10
20	73.10	75.17	64.83	68.97	60.69	71.72
25	73.10	74.48	64.83	68.97	61.38	73.79
30	73.10	74.48	64.83	68.97	61.38	73.79
35	73.10	73.79	64.83	68.97	60.69	73.79
40	73.10	73.79	64.83	68.97	62.07	73.79
45	73.10	73.79	64.83	68.97	62.07	73.79
50	73.10	73.79	64.83	68.97	62.07	73.79

Table 11: Classification accuracy after applying boosting on base classifiers using different ensemble size

Figure 17 compares the effect of applying bagging and boosting on each base classifier by plotting the accuracy of bagging and boosting with different ensemble size against the single classifier (i.e. without applying bagging and boosting) accuracy which is marked as a black horizontal line in the figures. It is observed that applying bagging and boosting on RBF and DT improved their accuracy in all different ensemble size. In addition, in case of MLP and NB, bagging produced higher accuracy than boosting. In case of RBF and DT, boosting accuracy was higher than bagging. In case of BBN, boosting accuracy was stable and bagging produced better accuracy than boosting as we increase the ensemble size, starting from ensemble size 30. In contrast, neither bagging nor boosting SVM improved its accuracy.

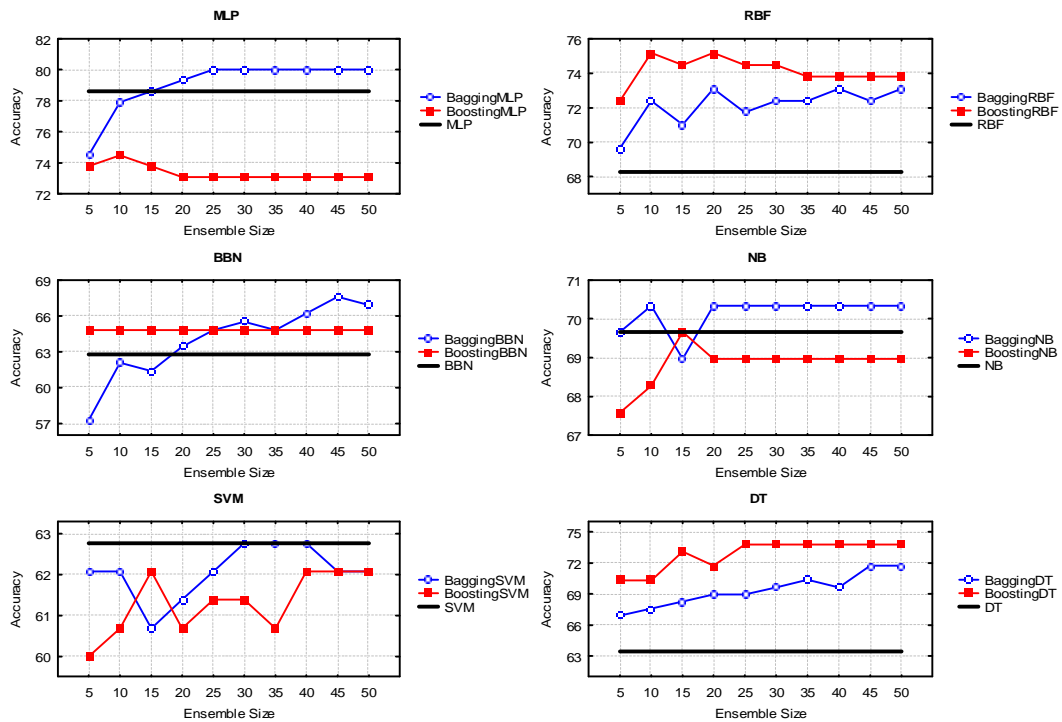


Figure 17: Single classifier Vs Bagging Vs Boosting

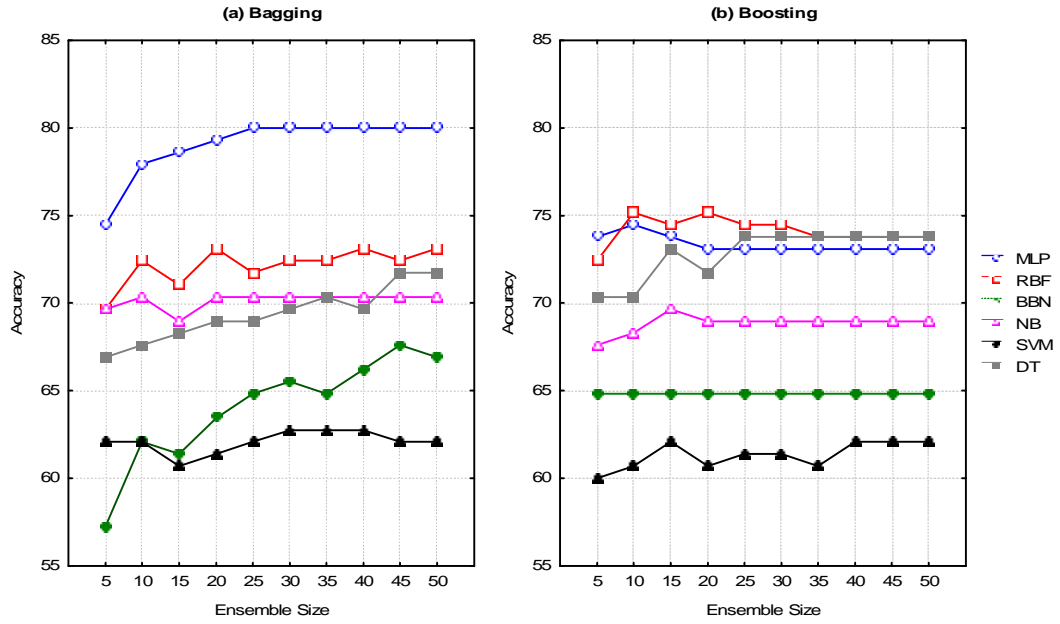


Figure 18: (a) Bagging all classifiers, (b) Boosting all classifiers

Figure 18 (a) and (b) plot the accuracy of bagging and boosting for each base classifier with different ensemble size, respectively. Figure 18 (a) indicates that bagging MLP improved its accuracy significantly and made it superior compared to the other models. Bagging DT improved its accuracy considerably and made it competitive with RBF and NB. However, even after applying bagging on SVM and BBN, they produced the worst accuracy. Figure 18 (b) indicates that boosting made MLP, RBF and DT close to each other to compete for the best accuracy; Boosting RBF and DT had a positive effect on these models accuracy, but boosting MLP decreased its accuracy. As in bagging, SVM and BBN produced the worst accuracy after applying boosting on them compared to the other boosted models.

There is a tradeoff between the incremental performance gains and the computational time that should be taken in consideration when applying ensemble techniques [6, 99]. Fortunately, we observed that when ensemble size was set to 25 and more, bagging and boosting did not produce significant different results over smaller ensemble sizes, i.e., most results are stable. As a result, we believe that ensemble size 25 is an appropriate value for ensemble size parameter in bagging and boosting for this dataset.

Figure 19 summarizes the comparison between the accuracy of single classifiers and their bagging and boosting when we set the ensemble size to 25. It can be observed that bagging improved the accuracy of 5 out of the 6 investigated models, whereas boosting improved the accuracy of only 3 out of the 6 models. In case of MLP and NB, the best accuracy was achieved by bagging. In case of RBF and DT, the best accuracy was achieved by boosting. In case of BNN, both bagging and boosting achieved the same accuracy. In case of SVM, neither bagging nor boosting improved its accuracy. Even after bagging and boosting, MLP remained as the best model in classification accuracy and boosting increased DT considerably and made it competitive with RBF for second highest.

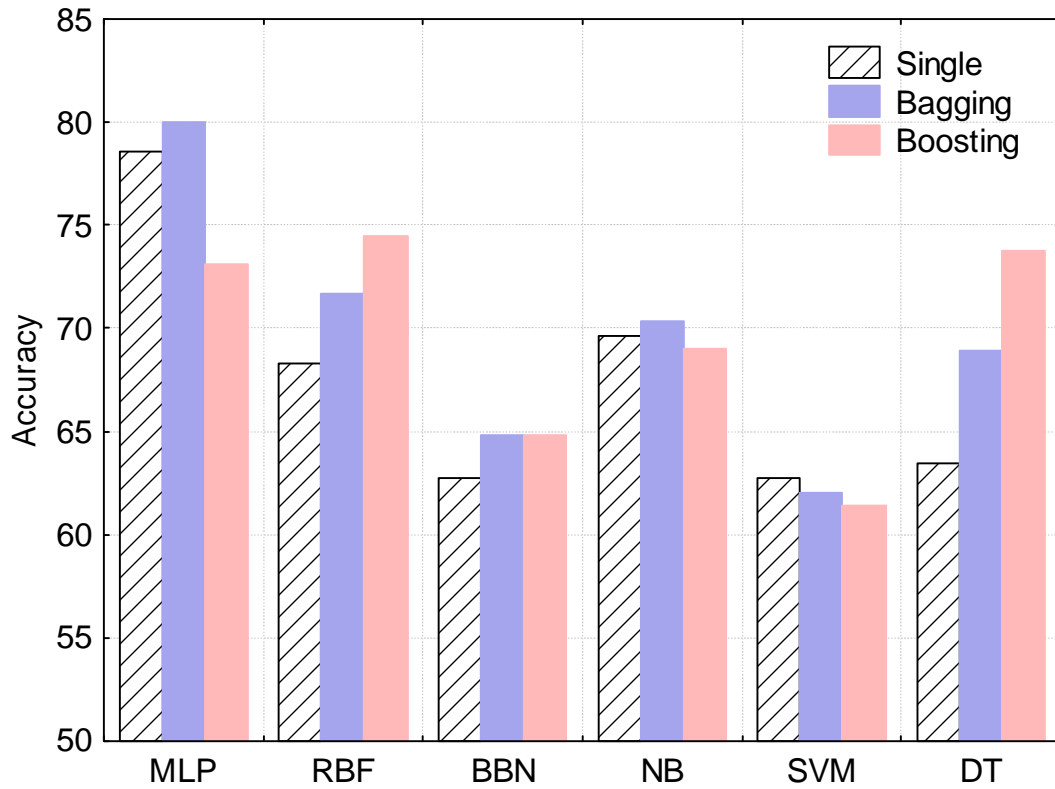


Figure 19: Single classifier Vs Bagging (25) Vs Boosting (25)

In summary of this chapter, empirical results indicate that bagging and boosting yield improved classification accuracy over most of the investigated single classifiers. However, bagging and boosting performance varied from one classifier to another. In some cases, bagging outperforms boosting, while in some other cases, boosting outperforms bagging. In case of MLP and NB, bagging produced the best accuracy. In case of RBF and DT, boosting produced the best accuracy. However, in case of SVM, bagging and boosting resulted in detrimental in accuracy. This result is supported by another study which states that SVM ensembles are not always better than a single SVM [99], which indicates that SVM is a stable method for the dataset in this experiment.

CHAPTER 5

Multi-model ensembles (Classification)

5.1 Multi-model ensembles for classification

In classification, a multi-model ensemble consists of a set of individually trained classifiers, whose predictions are combined into something called an arbitrator, which produce the final output [75]. Multi-model ensembles can be classified into linear and nonlinear ensembles based on the design of their arbitrator [58].

5.1.1 Linear ensembles

In linear ensembles, the arbitrator assigns prediction weights for each individual classifier, in a linear fashion as shown in Figure 20. There are a number of linear approaches to implement the linear architecture of the arbitrator. These approaches are majority voting, average probability, best probability, and weighted probability, as described in the following sections.

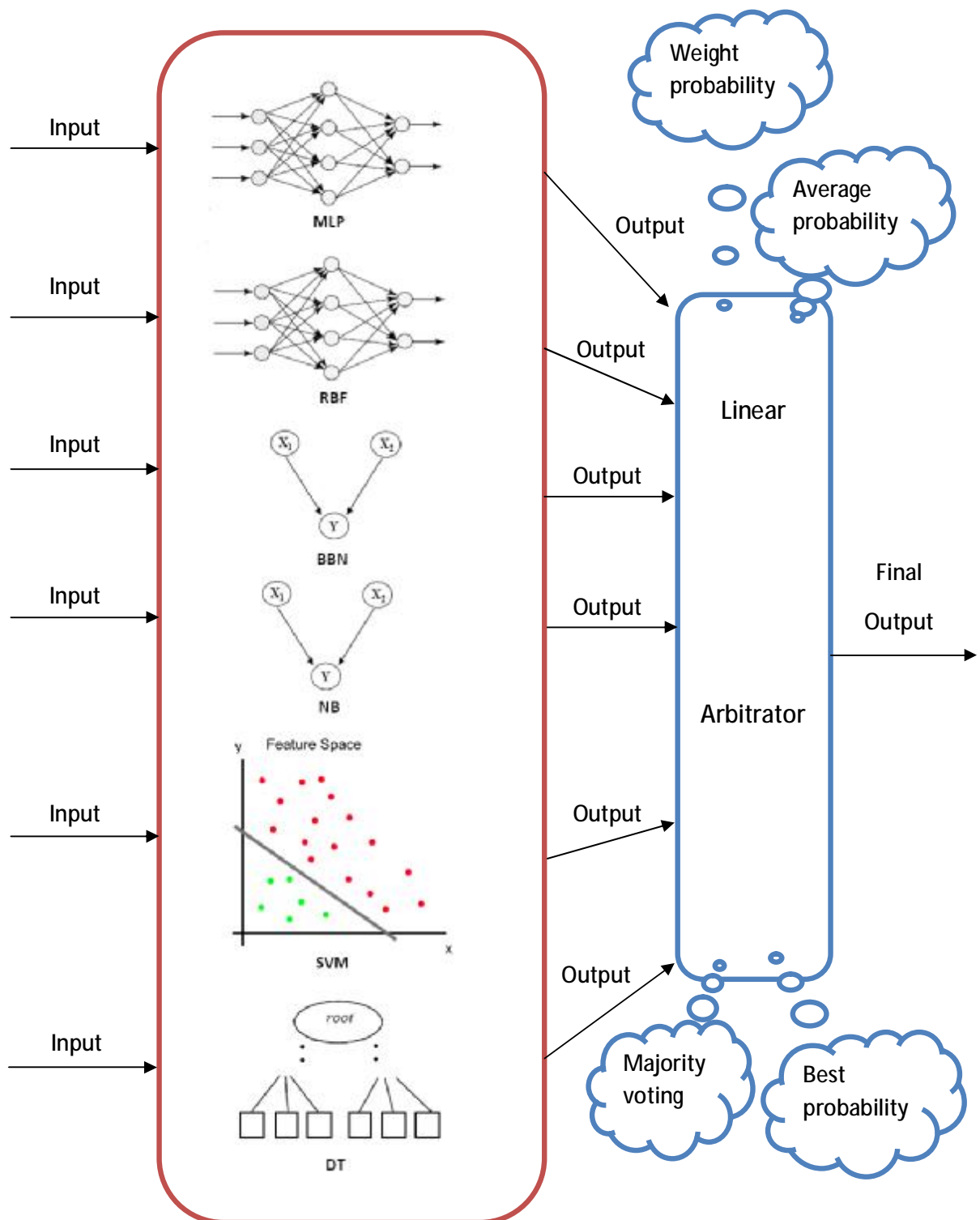


Figure 20: Multi-model linear ensemble (classification)

5.1.1.1 Majority voting

Majority voting ensemble assigns equal weight to the output of each prediction model. For each observation, the output classification (i.e. positive or negative) of each individual prediction model is taken as an input to the ensemble. Then, the majority voting ensemble will output the class (i.e. positive or negative), with the highest vote. However, since we have an even number of prediction models, an equal vote may be expected. In this situation, we will make the ensemble output the class as positive. Below figure, gives a formal description about this ensemble:

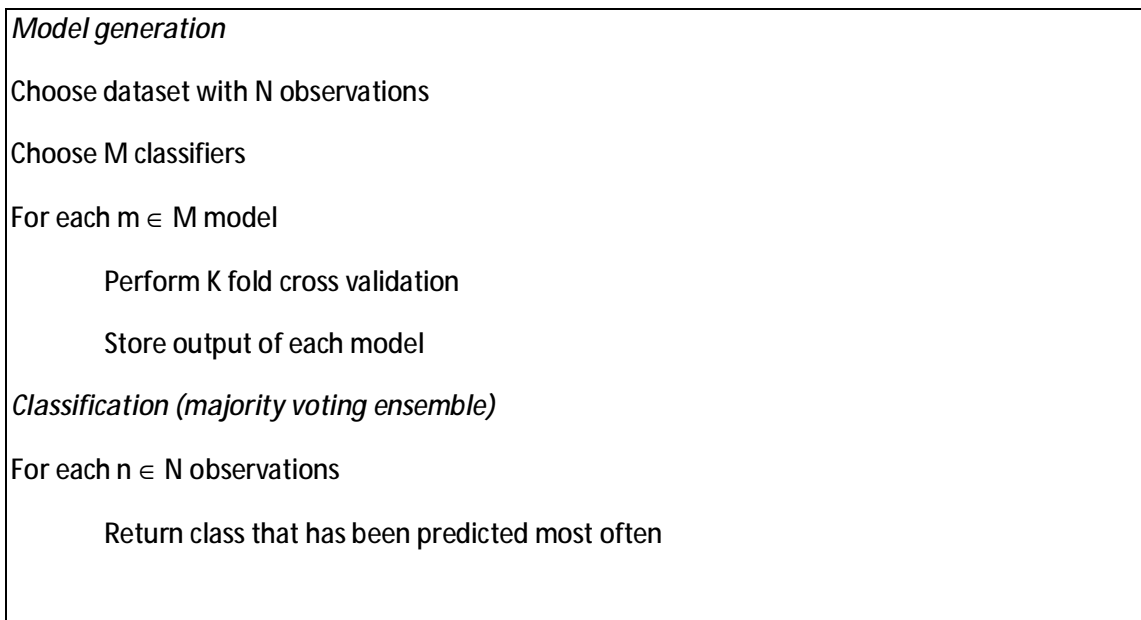


Figure 21: Majority voting linear ensemble

5.1.1.2 Average probability

Same as majority voting ensemble, each prediction model output has the same weight. For each observation, the probability values of the individual prediction models are taken as an input to the ensemble. Then, ensemble arbitrator outputs the average of these models probabilities. The following figure, gives a formal description about average ensemble:

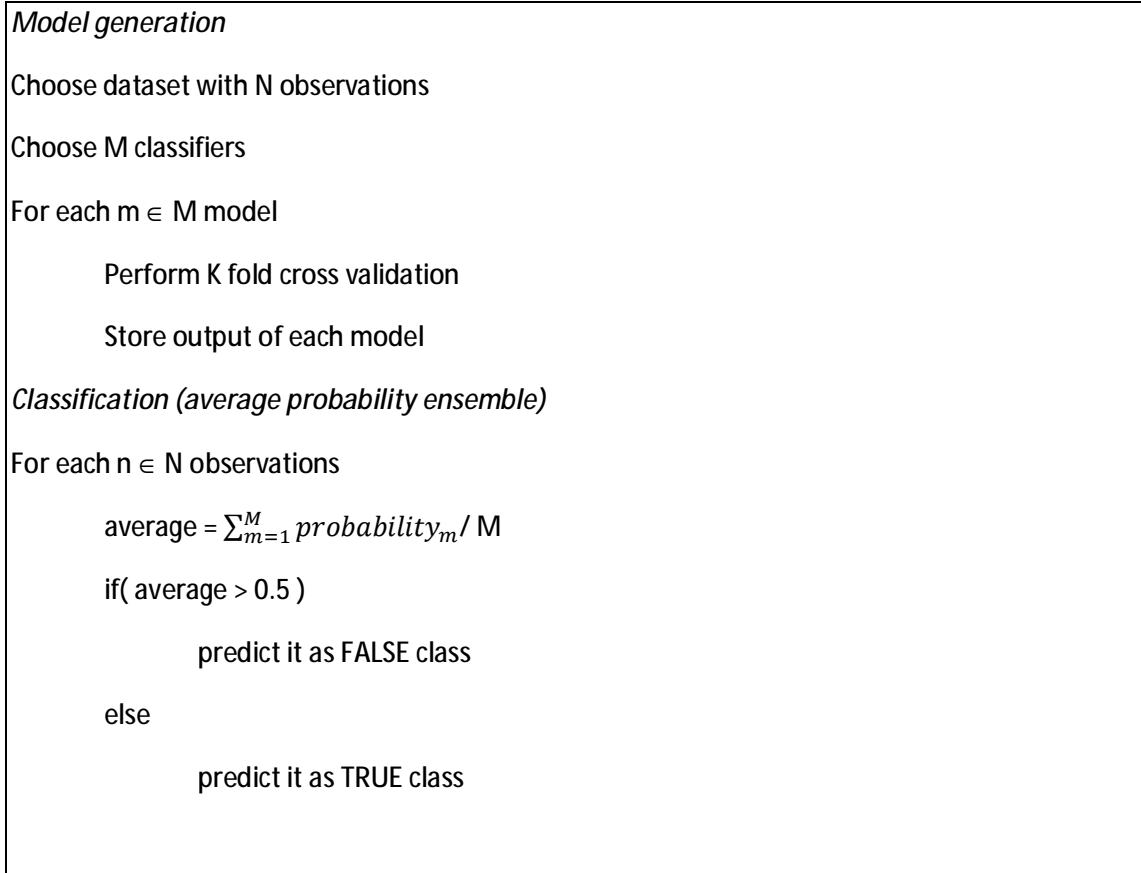


Figure 22: Average probability linear ensemble

5.1.1.3 Best probability

Best probability ensemble takes the advantage of the fact that classifiers have different errors across the used dataset partitions. The idea behind best probability ensemble is that across the dataset partitions, take the best trained classifier output performed in that partition. The following figure states the best linear ensemble approach:

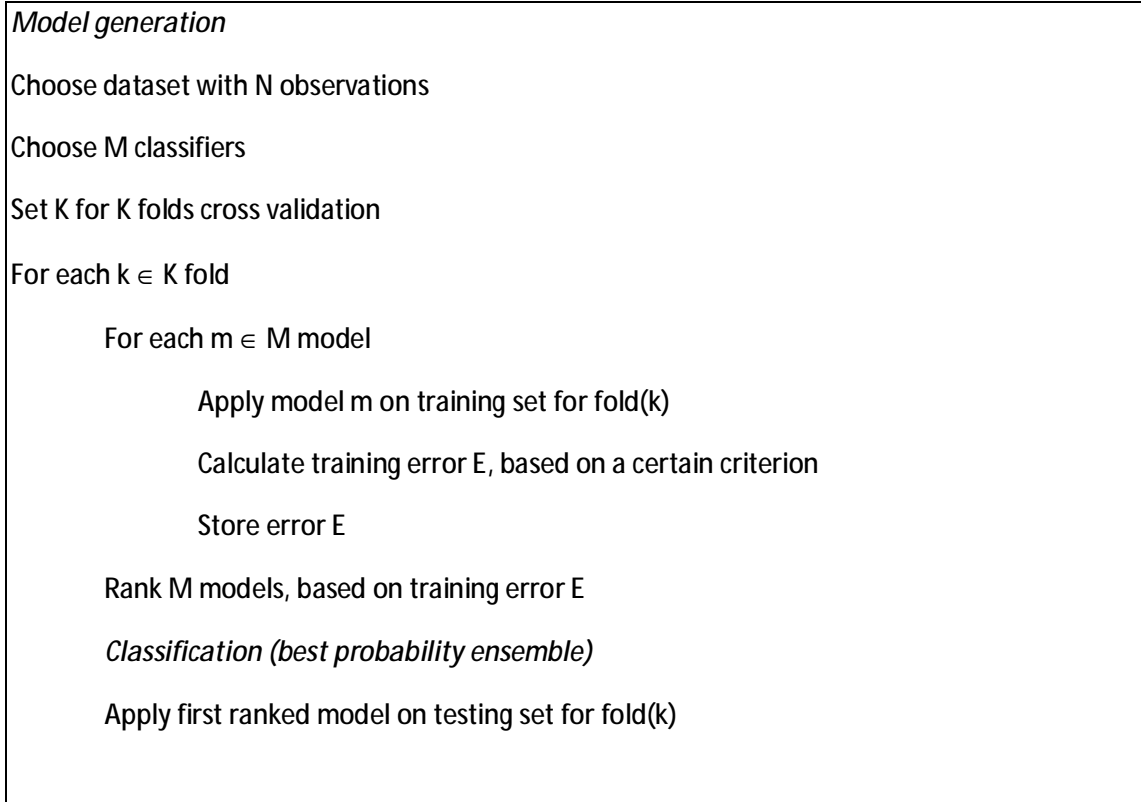


Figure 23: Best probability linear ensemble

5.1.1.4 Weighted probability

Weighted probability ensemble gives weight for individual models probability values, based upon a certain criterion. This criterion could be the accuracy evaluation measure. Prediction models with high error rate will be given less weight. The following figure, describes the formal approach of weight ensemble:

Model generation

Choose dataset with N observations

Choose M classifiers

Set K for K folds cross validation

For each $k \in K$ fold cross validation

 For each $m \in M$ models

 Apply m on training set for fold(k)

 Calculate training error E, based on a certain criterion

 Store error E

 Rank M models, based on training error E

 For each $m \in M$ model

 Apply model m on testing set fold(k)

 Multiply model m probability by its rank

 Store as probability(m)

Classification (weight probability ensemble)

For each $n \in N$ observations in fold(k)

$\text{Probability}_n = \sum_{m=1}^M \text{probability}_m / \sum_{i=1}^M i$

 if($\text{Probability}_n > 0.5$)

 predict as FALSE

 else

 predict as TRUE

Figure 24: Weight probability linear ensemble

5.1.2 Nonlinear ensembles

Nonlinear ensembles, as shown below, use nonlinear models to build its arbitrator. In arbitrator design, nonlinear model is used to assign the weights. Output from individual classifiers is fed to this nonlinear model as input to train it and assign weights accordingly. In our approach, we used the list of individual investigated models as our potential list of nonlinear arbitrators.

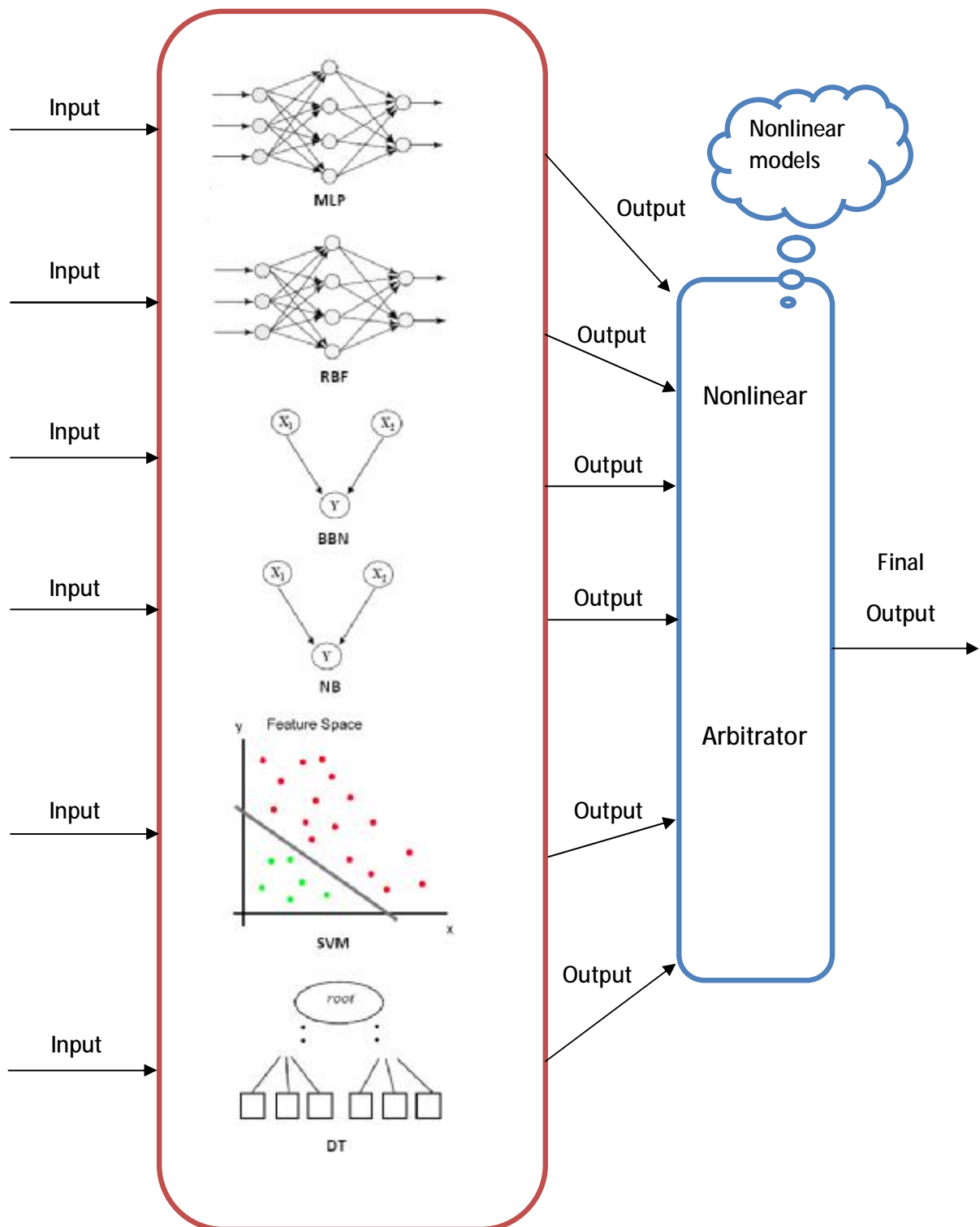


Figure 25: Multi-model nonlinear ensemble (classification)

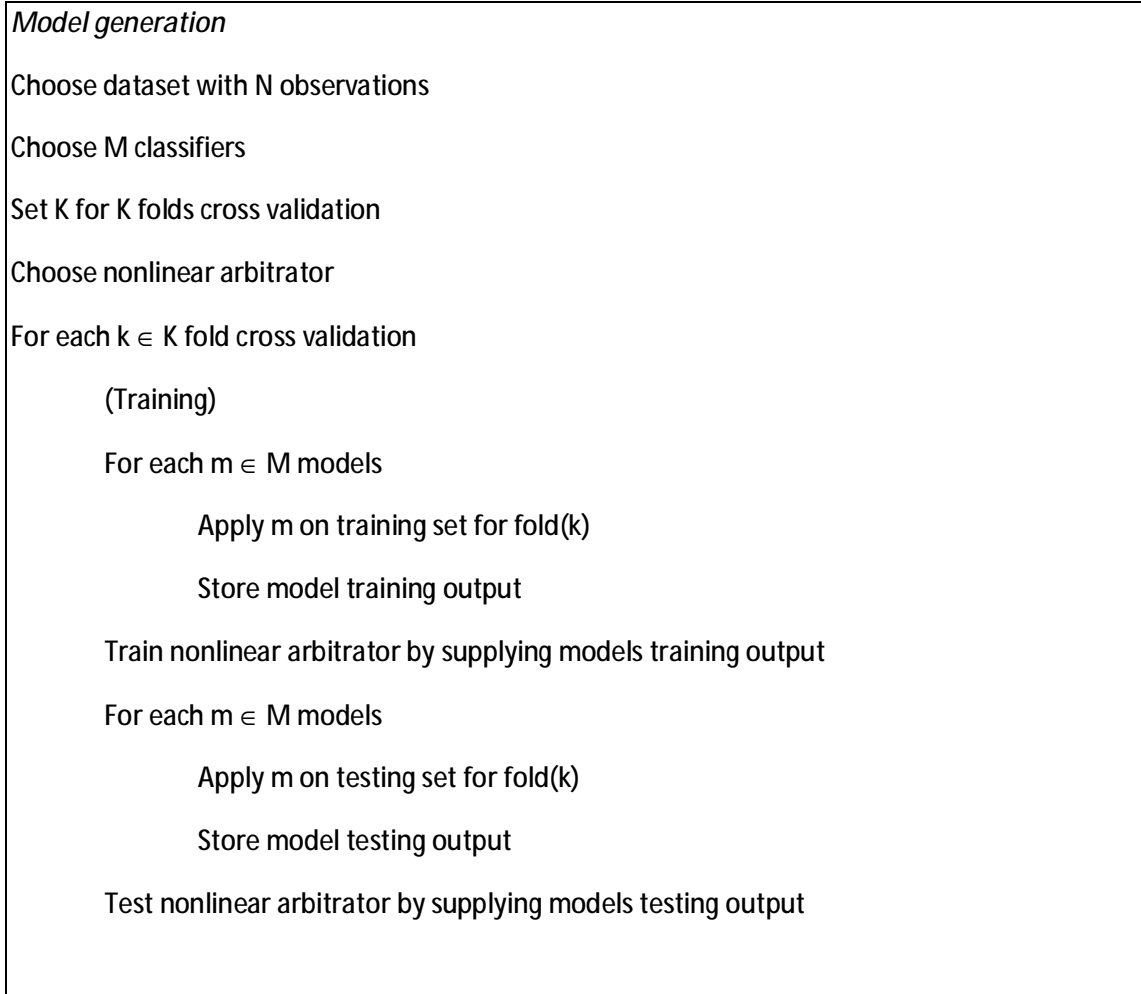


Figure 26: Nonlinear ensemble

5.2 Experiment design

5.2.1 Goal

This experiment incorporates many goals within it. The first goal is to examine the capabilities of individual classifiers. The second goal is to build different linear ensembles (i.e. majority voting, average probability, best probability, and weight probability) from chosen individual models, and examine to which extent they provide competitive results. The third goal is to build different nonlinear ensembles based on different nonlinear arbitrators (i.e. BBN, NB, MLP, RBF, SVM, and DT arbitrators), and

study their performance. Finally, we will compare the best model taken from each category (i.e. best model from individual models, linear ensembles, and nonlinear ensembles), and draw our conclusions.

5.2.2 Settings

The following table presents the settings for the conducted experiment:

Experiment type	Classification
Investigated models	BBN
	NB
	MLP
	RBF
	SVM
	DT
Data set	KC1
Evaluation measures	Accuracy, precision, recall, and f-measure
Validation technique	10 fold cross validation

Table 12: Experiment settings for multi-model ensemble (classification)

5.3 Results

Table 13 presents the results obtained from applying individual classifiers on KC1 dataset, with bold values indicating the best achieved result. Overall results show the superiority of MLP over all other models. MLP achieved the highest accuracy, with NB as second best. Moreover, there is a significant difference in accuracy, nearly 7%, between the highest accuracy (75.86%) achieved by MLP and the second highest accuracy (68.96%) achieved by NB. In the other hand, BBN produced the lowest accuracy.

In terms of precision, all models were competing for the best precision. However, MLP scored the highest precision value (0.67), with both SVM and NB as second highest, with no significant difference. Also, MLP scored the highest recall, and f-measure values (i.e. (0.81) and (0.73), respectively). However, SVM scored a very low results in the two measures (i.e. recall and f-measure) compared with other models.

Another observation from Table 13 is that SVM achieved a relatively moderate accuracy value, but a competing precision value. However, its performance, in terms of recall and f-measure, was the worst. This indicates that recall is encouraged to be used as a supporting measure for precision, and encourages the use of f-measure, since it represents the harmonic average of precision and recall.

	Individual models					
	BBN	NB	MLP	RBF	SVM	DT
Accuracy	58.62	68.96	75.86	63.44	61.37	65.51
Precision	0.5	0.64	0.67	0.58	0.64	0.59
Recall	0.66	0.56	0.81	0.38	0.15	0.53
F -measure	0.57	0.60	0.73	0.46	0.24	0.56

Table 13: KC1 results for individual models (classification)

Figure 27 plots individual classifiers ROC graph points, with different decision thresholds. It is observed that MLP curve intends to move to the northwest corner (i.e. best classifier). However, SVM was clearly the worst performed model, since its curve is the closest to southeast corner.

Table 14 presents the AUC values obtained from different individual models. MLP achieved the highest AUC value, and that means if other models curve intersect with MLP curve in some threshold, MLP will still give the best overall performance.

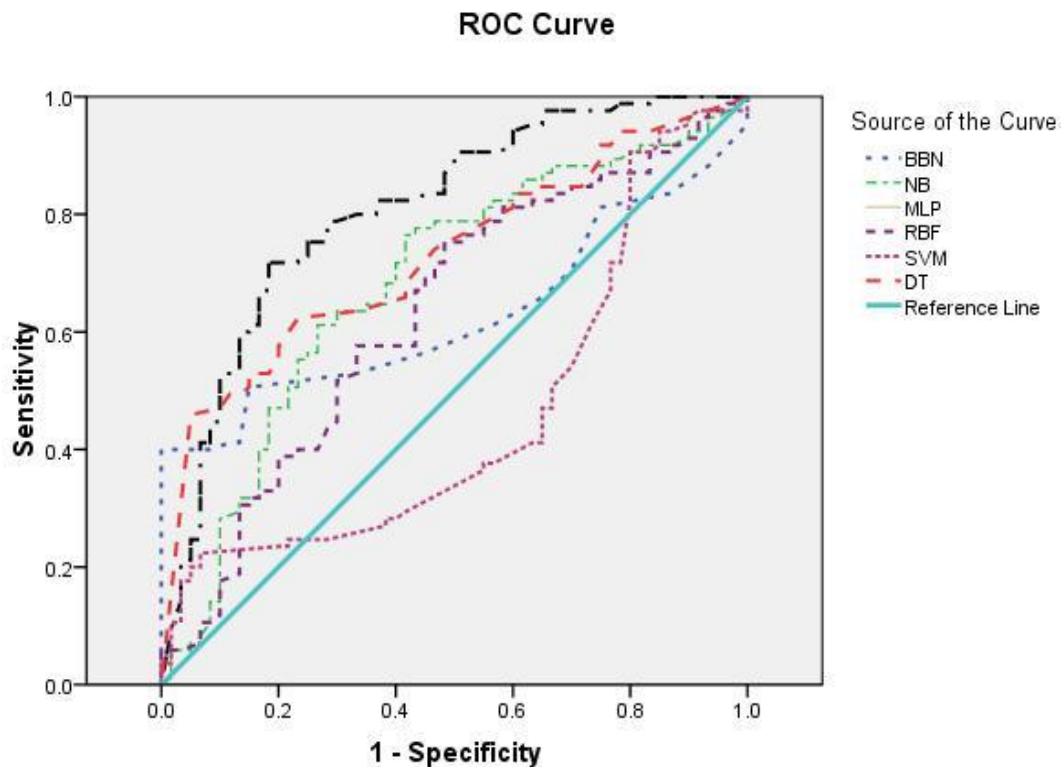


Figure 27: KC1 ROC graph for individual models (classification)

Individual models	Area Under Curve (AUC)
BBN	0.628
NB	0.683
RBF	0.631
SVM	0.459
DT	0.729
MLP	0.809

Table 14: Individual models AUC (KC1 classification)

Table 15 presents the results obtained from building different linear ensembles (i.e. majority voting, average probability, best probability, and weight probability), evaluated in the context of KC1 dataset. Weight ensemble achieved the highest accuracy, with majority voting and average ensembles competing for second highest. In terms of precision, recall, and f-measure, all linear ensembles were competing for the highest value, except for best ensemble, which scored the lowest values in all evaluation measures.

Best ensemble achieved the lowest accuracy among linear ensembles, because of its construction limitation. Best ensemble is based on the best trained classifier in each partition. In this case, SVM was the best trained classifier in most partitions, however, SVM was the one of worst in testing. This yielded in overall degraded accuracy of best ensemble.

	Linear Ensemble			
	Majority	Average	Best	Weight
Accuracy	71.03	69.65	60	73.79
Precision	0.64	0.64	0.55	0.68
Recall	0.68	0.6	0.16	0.68
F -measure	0.66	0.62	0.25	0.68

Table 15: KC1 results for linear ensembles (classification)

Figure 28 plots all linear ensemble models ROC graph points, except for majority voting ensemble, since it is not applicable. It is observed that both average and weight ensemble curves are competing for the best performance (i.e. curve more towards the northwest corner, and intersecting more often). However, from Table 16, we can conclude the weight probability ensemble is better, since it achieved the highest AUC value (0.775).

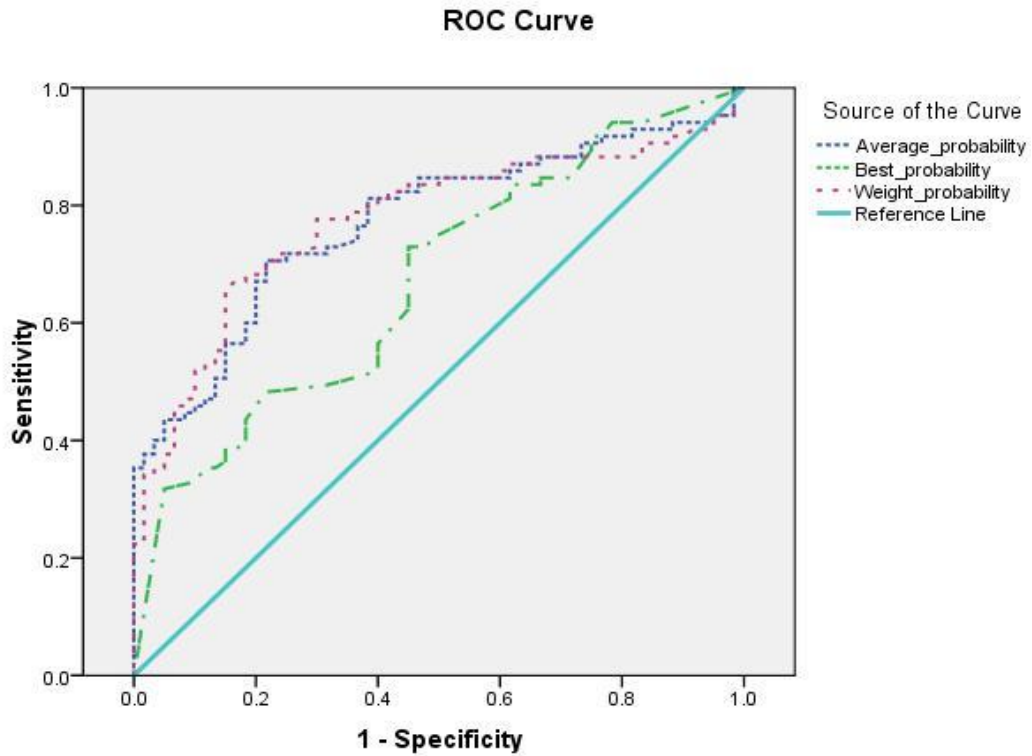


Figure 28: KC1 ROC graph for linear ensembles (classification)

Linear ensemble	Area Under Curve (AUC)
Average Probability	0.772
Best Probability	0.671
Weight Probability	0.775

Table 16: Linear ensemble models AUC (KC1 classification)

Table 17 presents the results obtained from building different nonlinear ensembles from different nonlinear arbitrators (i.e. BBN, NB, MLP, RBF, SVM, and DT), evaluated in the context of KC1 dataset. Overall results show the superiority of SVM nonlinear ensemble over all other nonlinear ensembles. SVM ensemble achieved the highest accuracy, precision, recall, and f-measure, with DT as second best in all measures.

	Nonlinear Ensemble					
	BBN	NB	MLP	RBF	SVM	DT
Accuracy	71.03	71.03	69.65	69.65	75.86	72.41
Precision	0.64	0.64	0.62	0.62	0.67	0.65
Recall	0.68	0.68	0.65	0.65	0.81	0.71
F -measure	0.66	0.66	0.63	0.63	0.73	0.68

Table 17: KC1 results for nonlinear ensembles (classification)

Figure 29 plots all nonlinear ensemble models ROC graph points, with different decision thresholds. It is observed from ROC graph, that nonlinear ensembles compete for best model, as we change threshold. In addition, Table 18 gives close AUC values between different nonlinear ensembles. However, both MLP and NB nonlinear ensembles achieved the highest AUC value (0.744).

In general, we rely on our main evaluation measures (i.e. accuracy, precision, recall, and f-measure), to come with our best performed nonlinear ensemble. So, we conclude that SVM ensemble is the best performed nonlinear ensemble.

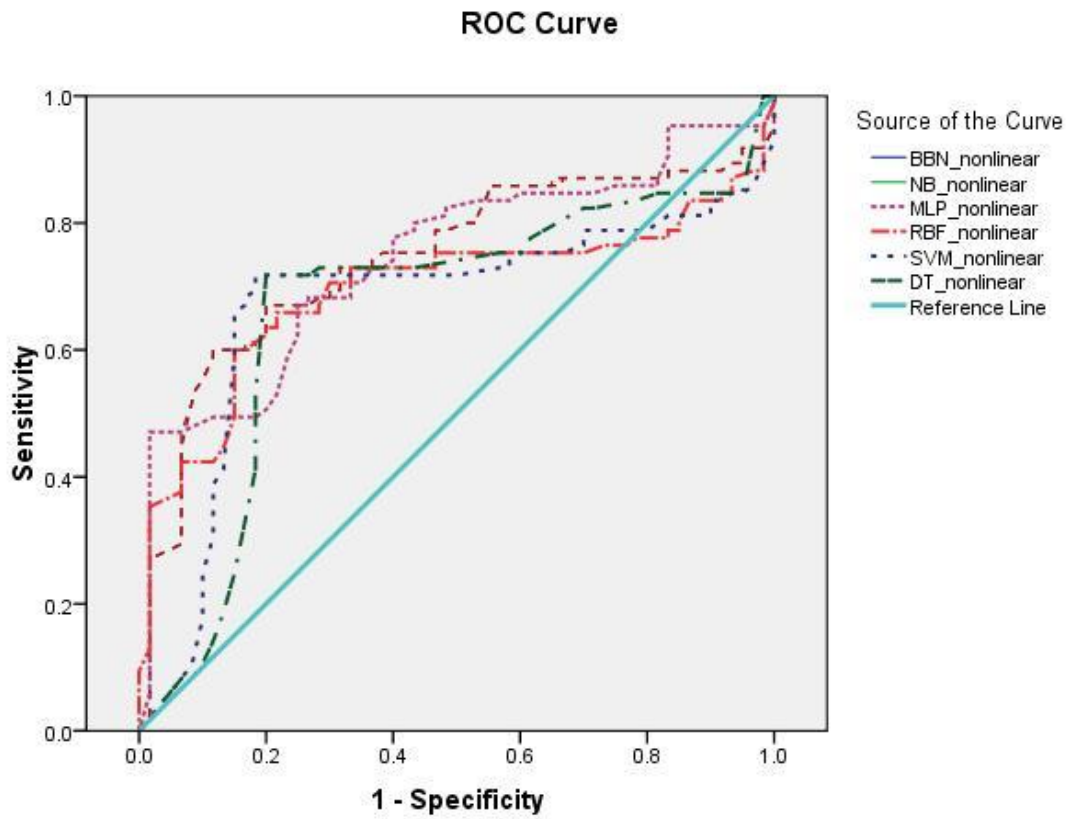


Figure 29: KC1 ROC graph for nonlinear ensembles (classification)

Linear ensemble	Area Under Curve (AUC)
BBN	0.743
NB	0.744
MLP	0.744
RBF	0.694
SVM	0.670
DT	0.66

Table 18: Nonlinear ensemble models AUC (KC1 classification)

Now, we discussed the performance of various models in each category (i.e. individual, linear ensemble, nonlinear ensemble models). Next, we pick the best model from each category as Table 19 shows, and compare them, to examine to which extent ensembles offer an increase in performance. MLP was the best model as an individual model, weight ensemble as linear ensemble, and SVM ensemble as nonlinear ensemble.

Table 19 shows that weight ensemble was slightly lower than MLP model in accuracy, and f-measure, and slightly higher in precision. However, SVM nonlinear ensemble produced competitive results against MLP.

	Individual model	Linear ensemble	Nonlinear ensemble
	MLP	Weight	SVM
		Accuracy	
Accuracy	75.86	73.79	75.86
Precision	0.67	0.68	0.67
Recall	0.81	0.68	0.81
F -measure	0.73	0.68	0.73

Table 19: KC1 comparison of individual Vs best linear Vs best nonlinear (classification)

From figure and table below, MLP curve competed with weight linear ensemble curve for best model performance. However, MLP scored higher AUC value than weight linear ensemble, with a difference of (0.034). Based on results in Table 19, we concluded that ensembles in general offer competitive results against individual models.

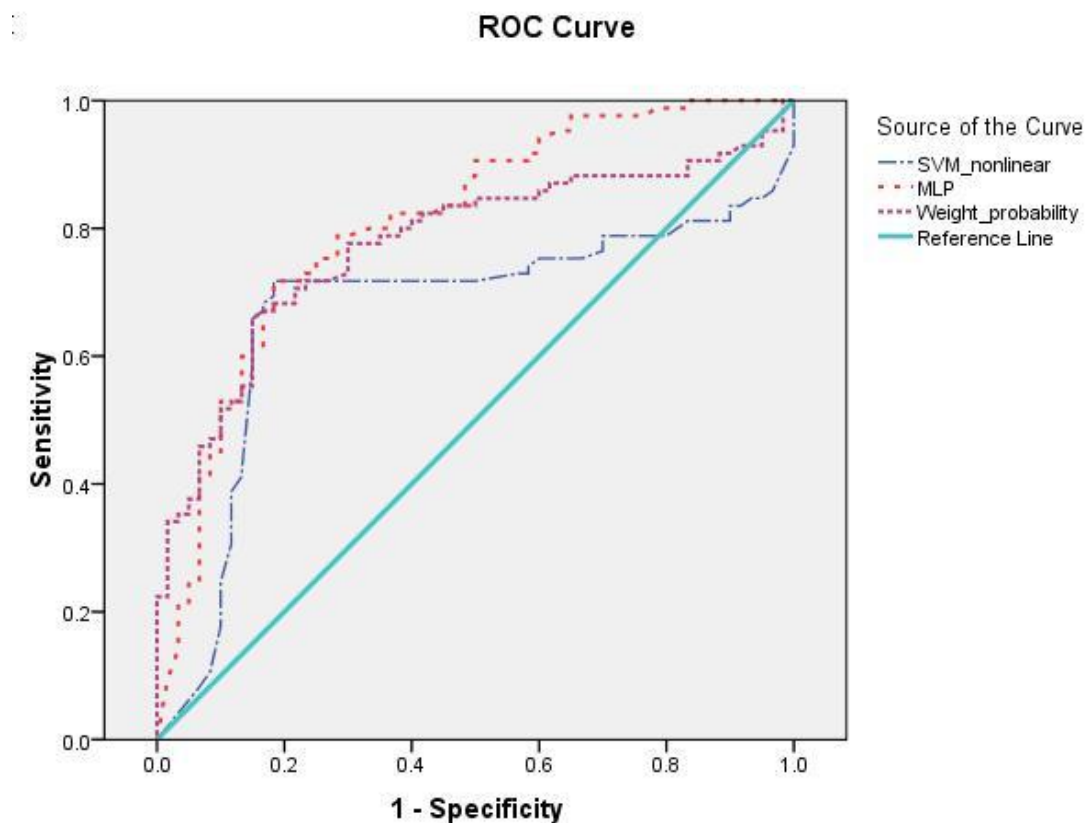


Figure 30: KC1 ROC graph for comparison of individual Vs best linear Vs best nonlinear (classification)

Model	Area Under Curve (AUC)
MLP	0.809
Weight probability	0.775
SVM nonlinear	0.670

Table 20: Comparison of individual Vs best linear Vs best nonlinear AUC (classification)

CHAPTER 6

Multi-model ensembles (Regression)

6.1 Multi-model ensembles for regression

In regression, a multi-model ensemble consists of a set of individually trained regression models, whose predictions are combined into something called an arbitrator, which produce the final output [75]. Multi-model ensembles can be classified into linear and nonlinear ensembles based on the design of their arbitrator [58].

6.1.1 Linear ensembles

In linear ensembles, the arbitrator assigns prediction weights for each individual regression model, in a linear fashion as shown in Figure 31. There are a number of linear approaches to implement the linear architecture of the arbitrator. These approaches are average, best, and weight, as described in the following sections.

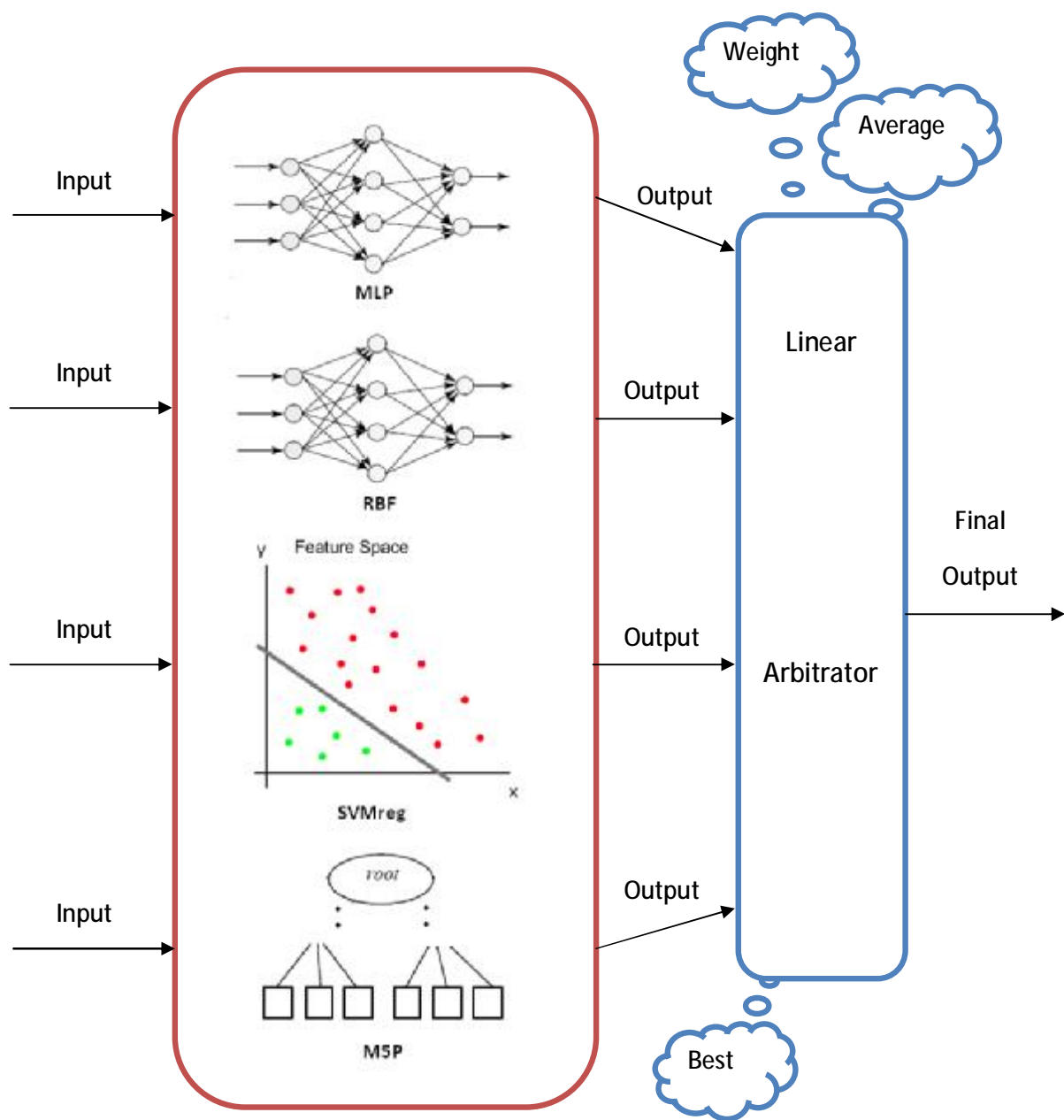


Figure 31: Multi-model linear ensemble (regression)

6.1.1.1 Average

Average linear ensemble is the simplest ensemble model, where each prediction model has the same weight. For each observation, the output values of the individual prediction models are taken as input to the ensemble and the average of these values as the output by the ensemble. The following figure, gives a formal description about average ensemble:

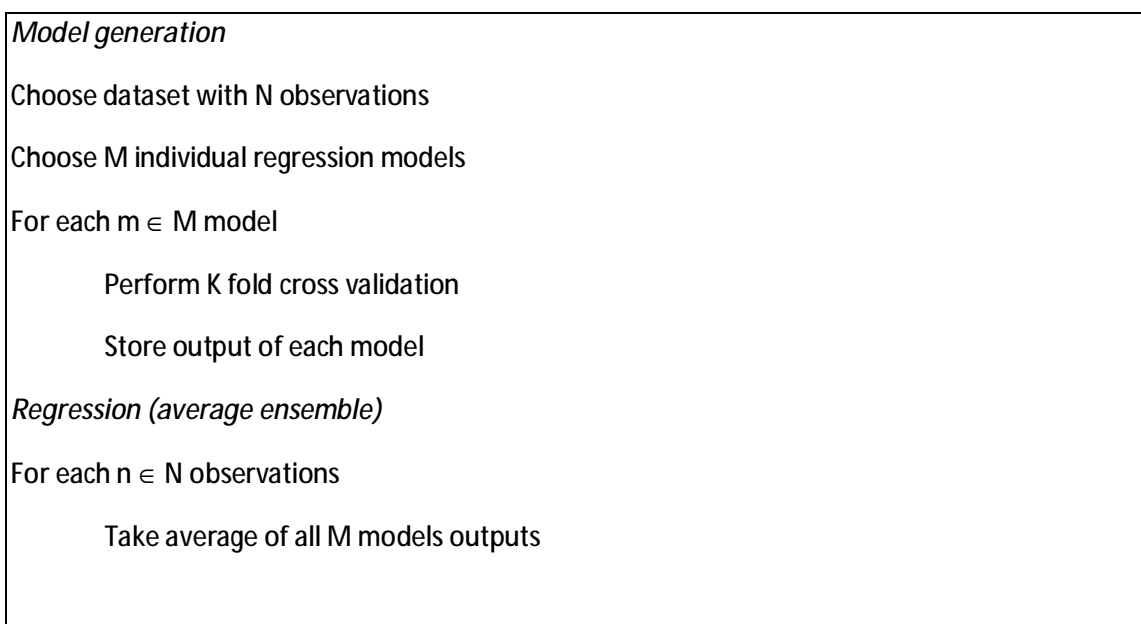


Figure 32: Average linear ensemble

6.1.1.2 Best

Best linear ensemble takes the advantage of the fact that regression models have different errors across the used dataset partitions. The idea behind best linear ensemble is that across the dataset partitions, take the best trained regression model performed in that partition. The following figure states the best linear ensemble approach:

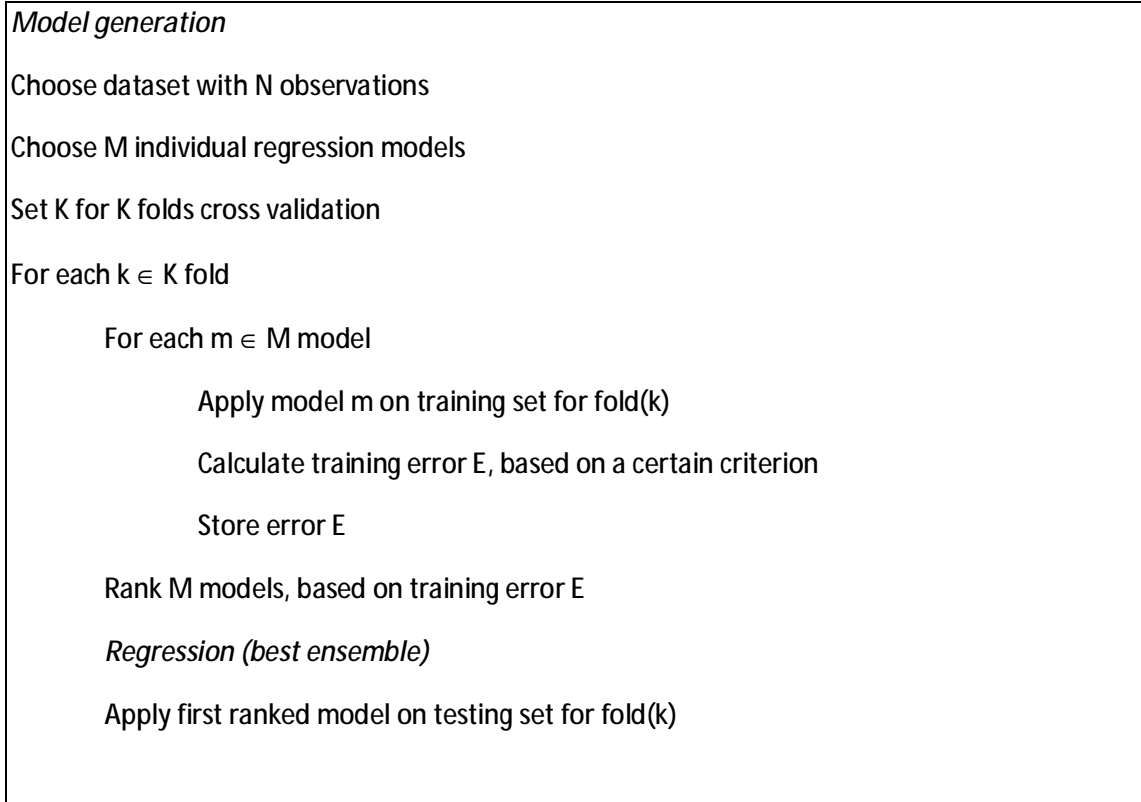


Figure 33: Best linear ensemble

6.1.1.3 Weight

Individual output values are given weights based upon a certain criterion. This criterion could be mean magnitude of relative error (MMRE). Prediction models with high error rate will be given less weight. The following figure, describes the formal approach of weight ensemble:

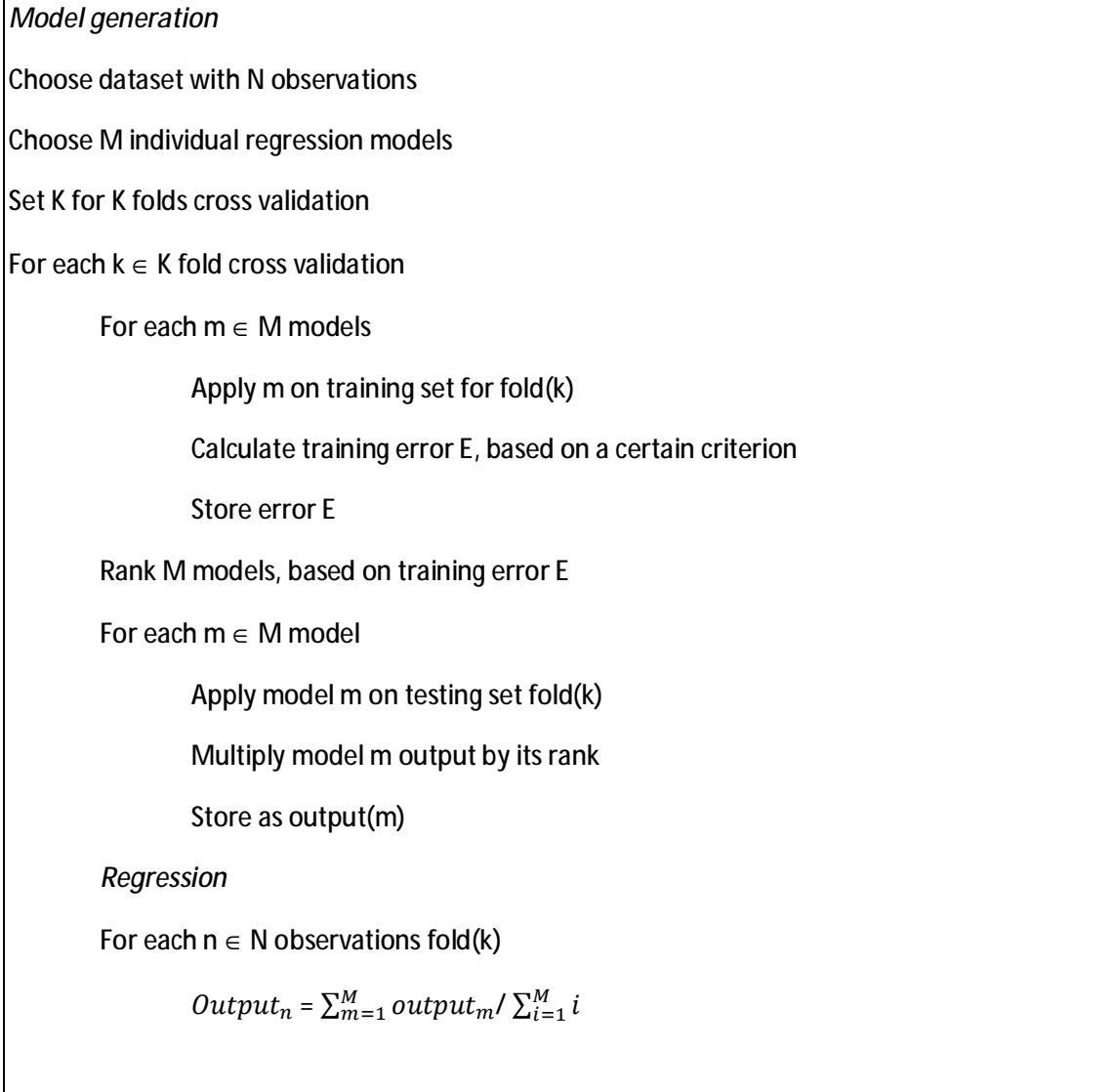


Figure 34: Weight linear ensemble

6.1.2 Nonlinear ensembles

Nonlinear ensembles, as shown below, use nonlinear models to build its arbitrator. In arbitrator design, nonlinear model is used to assign the weights. Output from individual regression models is fed to this nonlinear model as input to train it and assign weights accordingly. In our approach, we used the list of individual investigated models as our potential list of nonlinear arbitrators.

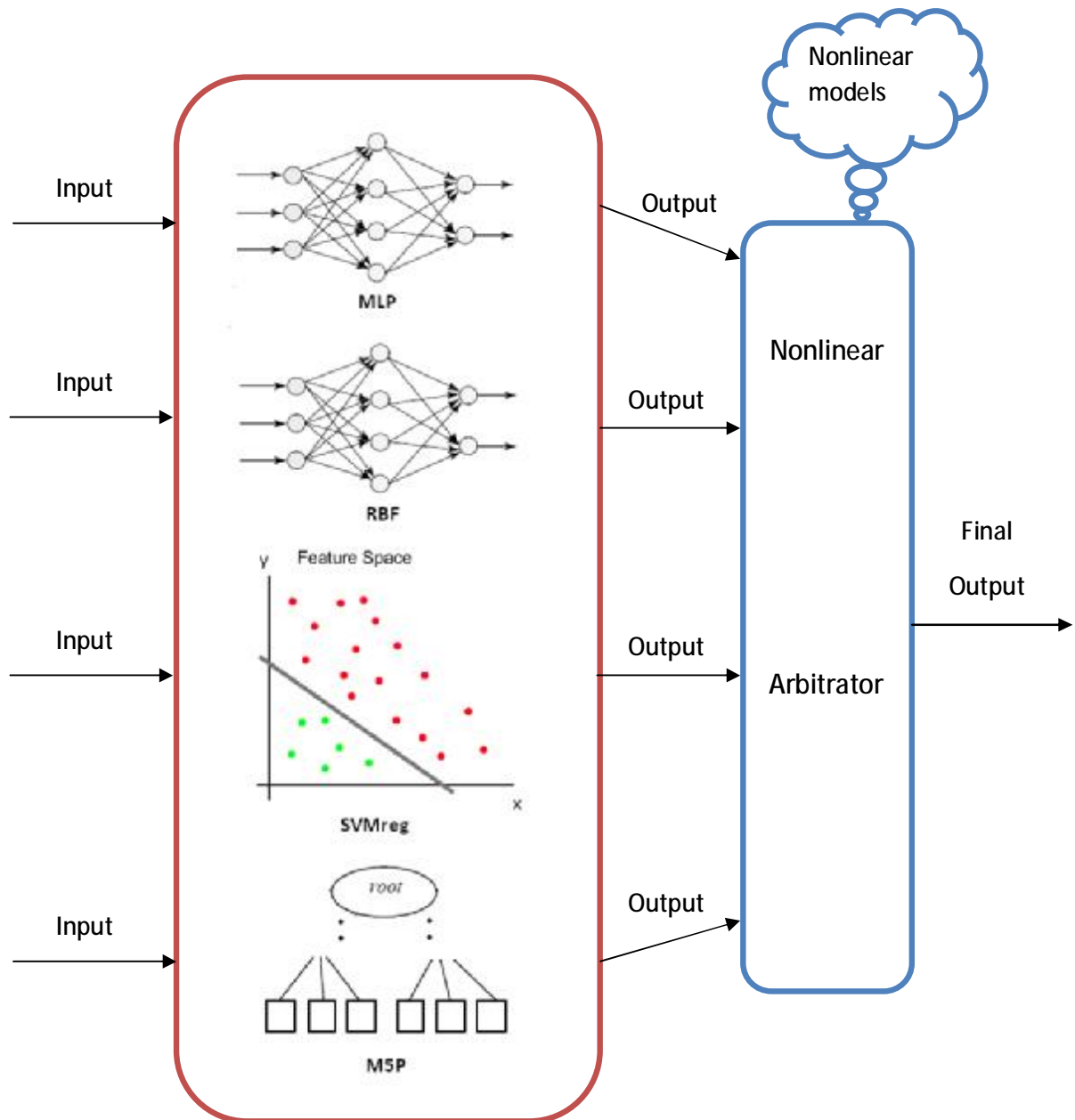


Figure 35: Multi-model nonlinear ensemble (regression)

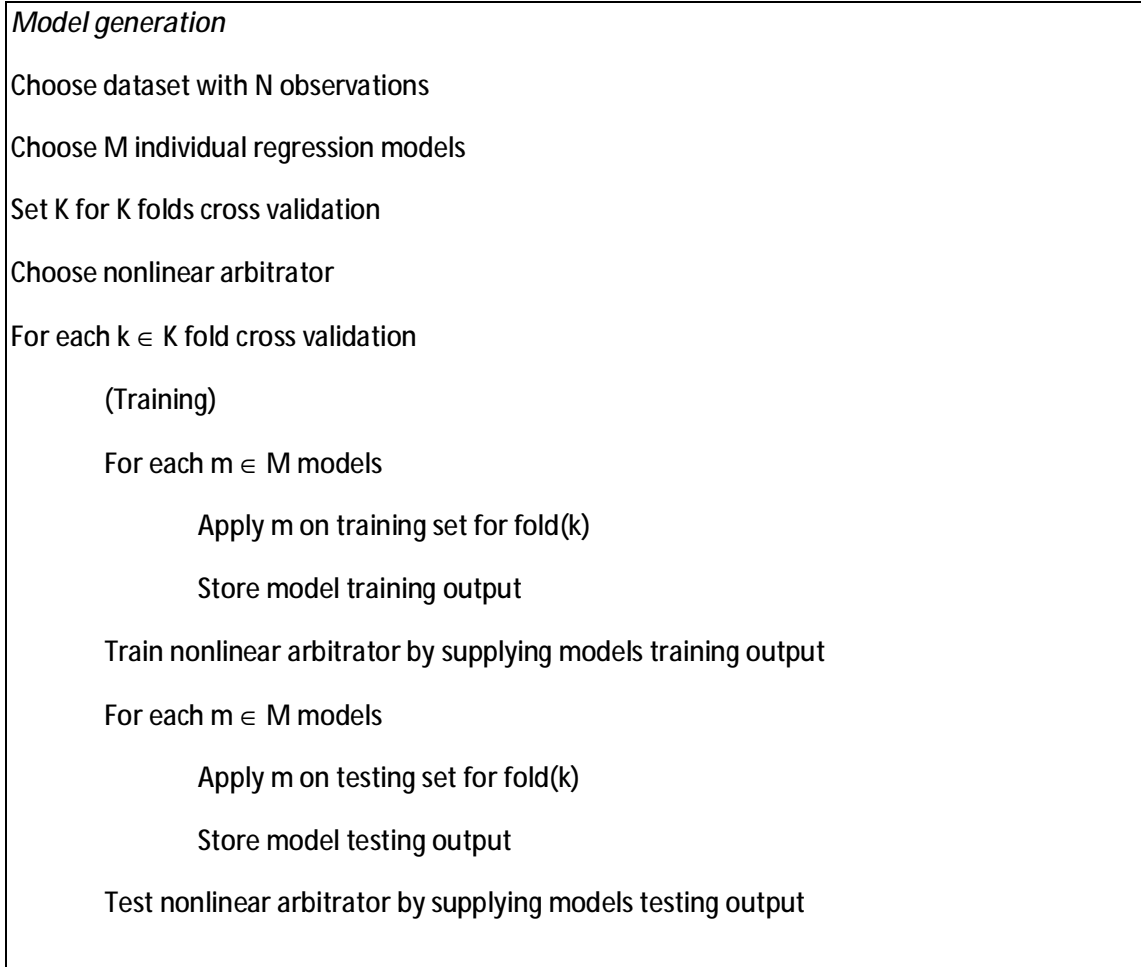


Figure 36: Nonlinear ensemble

6.2 Experiment design

6.2.1 Goal

This experiment has many goals to achieve. The first goal is to examine the capabilities of individual regression models. The second goal is to build different linear ensembles (i.e. average, best, and weight) from chosen individual models, and examine to which extent they provide competitive results. The third goal is to build different nonlinear ensembles based on different nonlinear arbitrators (i.e. MLP, RBF, SVMreg, and M5P arbitrators), and study their performance. Finally, we will compare the best model taken

form each category (i.e. best model from individual models, linear ensembles, and nonlinear ensembles), and draw our conclusions.

6.2.2 Settings

In this experiment, we used three datasets KC1, UIMS, and QUES. KC1 is from the class fault domain, while the other two (i.e. UIMS and QUES) taken from the maintainability domain. The following table presents the settings for the conducted experiment:

Experiment type	Regression
Investigated models	MLP
	RBF
	SVMreg
	M5P
Data set	KC1, UIMS, and QUES
Evaluation measure	MMRE, StdMRE, and Pred(0.3)
Validation technique	10 fold cross validation

Table 21: Experiment settings for multi-model ensemble (regression)

6.3 Results

6.3.1 KC1

Table 22 presents the results obtained from applying individual prediction models on KC1 dataset, with bold values as indication of best achieved result. Overall results show a competition between RBF and SVMreg. SVMreg achieved the best MMRE, and RBF was second best with a minor MMRE difference of (0.06). However, RBF was better than MLP in StdMRE with (0.62) difference. In Pred(0.3), SVMreg and M5P got the best result. The worst model in performance was MLP (i.e. worst in MMRE, and Pred(0.3)).

	Individual models			
	MLP	RBF	SVMreg	M5P
MMRE	2.19	1.68	1.62	1.91
StdMRE	2.73	2.68	3.30	3.98
Pred(0.3)	20	21.66	23.33	23.33

Table 22: KC1 results for individual models (regression)

Figure 37 box plots KC1 results obtained from individual regression models. RBF was the best model, since it had the narrowest box and smallest whisker. However, SVMreg was competitive with RBF in the lower box. The worst model was MLP (i.e. biggest box and longest whisker).

In summary, RBF and SVMreg showed close and competitive results, but superior to other models. We chose SVMreg as best individual regression model, since it achieved the best MMRE and Pred(0.3).

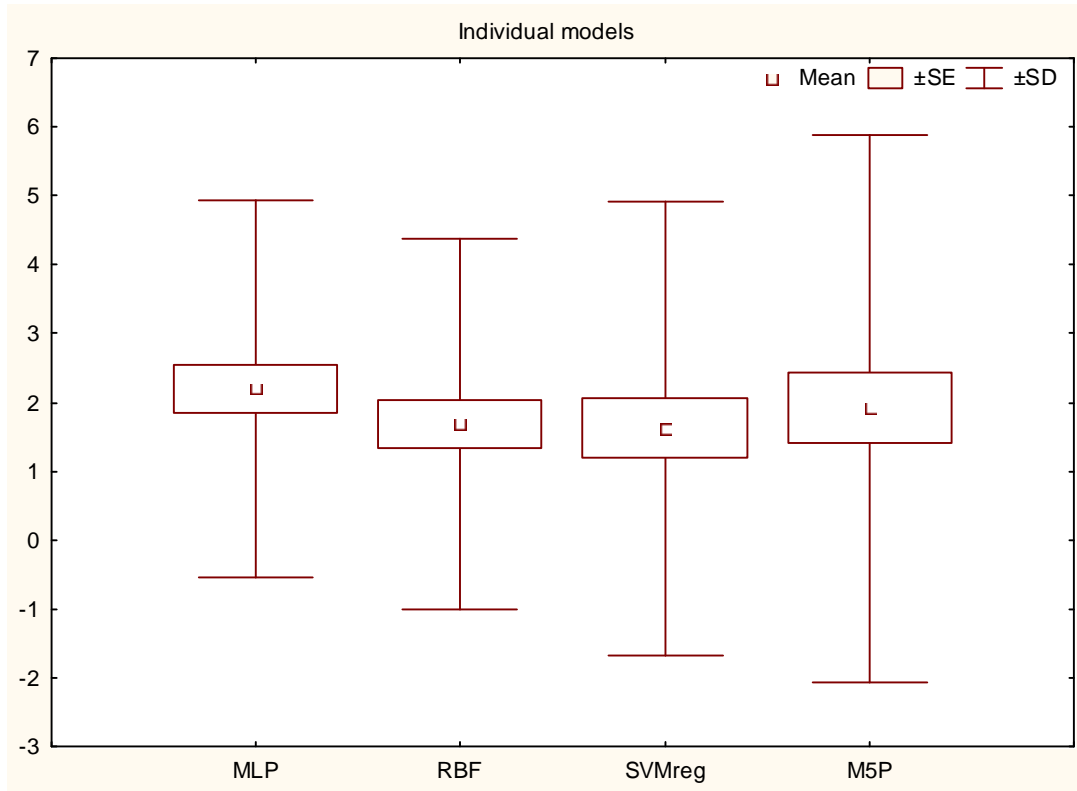


Figure 37: KC1 box plots for individual models (regression)

Table 23 presents the results obtained from building different linear ensembles (i.e. average, best, and weight). Linear ensembles show close and competitive results. Weight ensemble achieved the lowest MMRE, but the difference between it and average and best ensembles was (0.02) and (0.05) respectively. Also, weight ensemble achieved the lowest in StdMRE, with minor difference than second lowest (i.e. average ensemble) around (0.1). In terms of Pred(0.3), weight and average ensembles got the best results.

	Linear Ensemble		
	Average	Best	Weight
		MMRE	MMRE
MMRE	1.74	1.77	1.72
StdMRE	3.05	3.39	2.95
Pred(0.3)	26.66	25	26.66

Table 23: KC1 results for linear ensembles (regression)

Figure 38 box plots KC1 results obtained from created linear ensemble models. Boxes sizes seem to be close to each other, and they are at the same level. However, weight ensemble had the smallest whisker compared with others.

In summary, results were close and competitive. However, weight ensemble was slightly better than average and best ensembles.

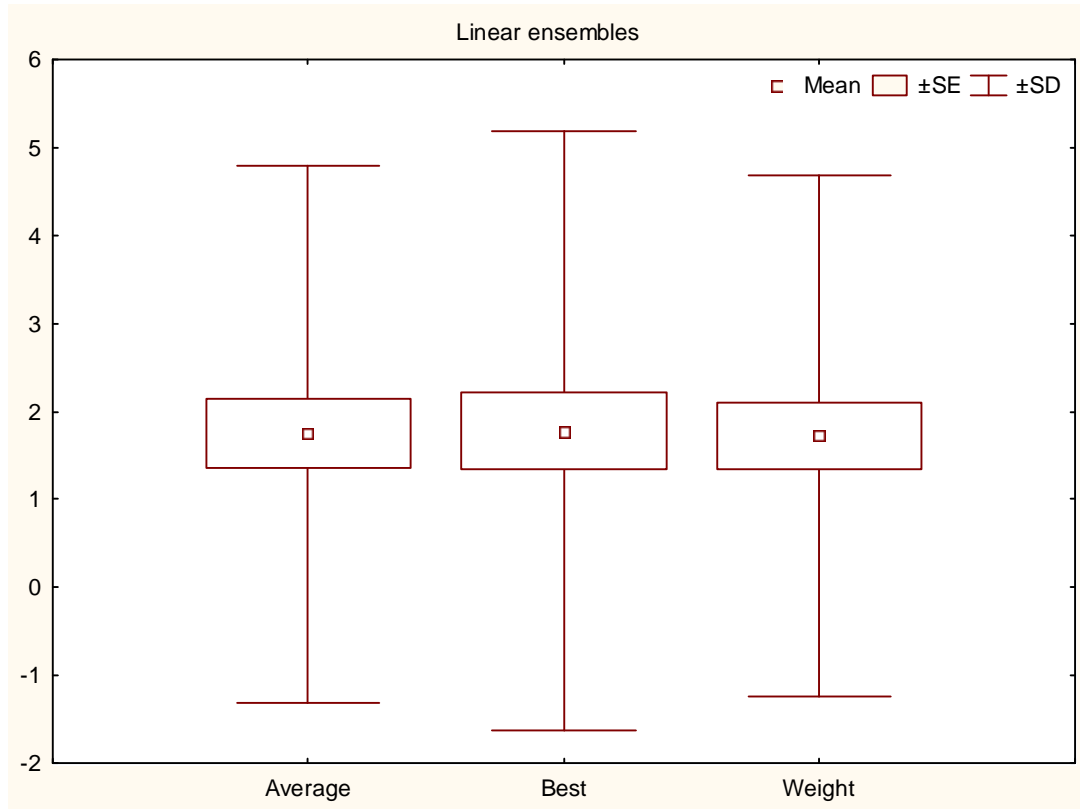


Figure 38: KC1 box plots for linear ensembles (regression)

Table 24 presents the results obtained by building nonlinear ensembles using nonlinear arbitrators (i.e. different nonlinear models). SVMreg nonlinear ensemble outperformed all other nonlinear ensembles in all performance measures. RBF nonlinear ensemble came second best, with minor difference between it and best nonlinear ensemble (i.e. SVMreg nonlinear ensemble), in terms of MMRE and StdMRE, (0.03) and (0.3), respectively. However, in Pred(0.3) SVMreg nonlinear ensemble was superior over other nonlinear ensembles with Pred(0.3) value of 26.6%.

	Nonlinear Ensemble			
	Nonlinear models			
	MLP	RBF	SVMreg	M5P
MMRE	2.29	1.70	1.67	2.12
StdMRE	4.32	2.66	2.35	2.77
Pred(0.3)	16.66	20	26.66	15

Table 24: KC1 results for nonlinear ensembles (regression)

Figure 39 box plots KC1 results obtained from created nonlinear ensemble models. SVMreg nonlinear ensemble had the narrowest box, smallest whisker, and the lowest box and whisker among the other models, which means that it's the best nonlinear ensemble. On the other hand, MLP nonlinear ensemble was the worst (i.e. biggest box and longest whisker).

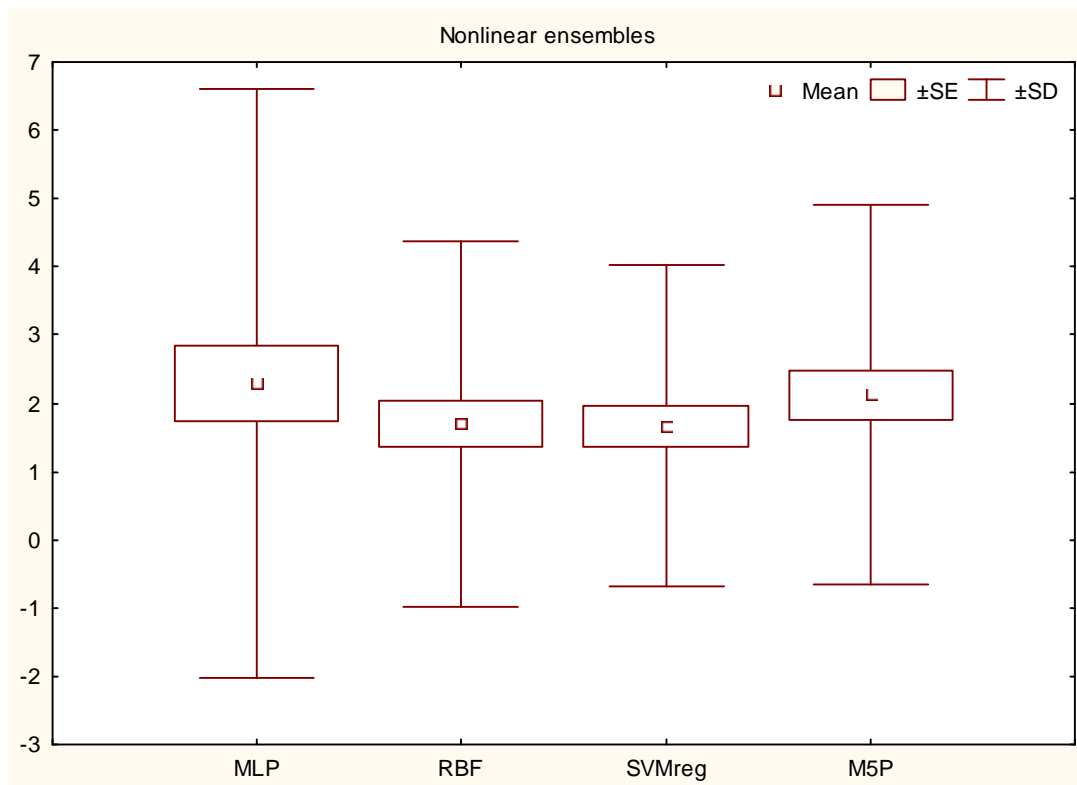


Figure 39: KC1 box plots for nonlinear ensembles (regression)

Now, we discussed the performance of various models in each category (i.e. individual, linear ensemble, nonlinear ensemble models). Next, we pick the best model from each category as Table 25 shows, and compare them, to examine to which extent ensembles offer an increase in performance. SVMreg was the best model as an individual model, weight ensemble as linear ensemble, and SVMreg ensemble as nonlinear ensemble.

As Table 25 shows, best individual model achieved the lowest MMRE, with a minor difference between it and second lowest MMRE (i.e. nonlinear ensemble) around (0.05). However, both ensembles were better than individual in StdMRE and Pred(0.3). In overall, nonlinear ensemble outperformed other models, since it shows competitive results.

Moreover, we performed Wilcoxon significance test at p-level equal to (0.1), to examine the MRE significance difference between the best model in each category. Table 26 shows the results in terms of p-value (upper cell) and z-value (lower cell). Bold values indicate significant difference. Results show that the difference between ensembles (i.e. linear and nonlinear) and SVMreg was significant. However, the difference between linear and nonlinear ensemble was not significant.

	individual	Linear	Nonlinear
		Weight	
	SVMreg	MMRE	SVMreg
MMRE	1.62	1.72	1.67
StdMRE	3.30	2.95	2.35
Pred(0.3)	23.33	26.66	26.66

Table 25: KC1 comparison of individual Vs best linear Vs best nonlinear (regression)

	SVMreg	Weight linear ensemble	SVMreg nonlinear ensemble
SVMreg			
Weight linear ensemble	0.057		
	1.907		
SVMreg nonlinear ensemble	0.035	0.740	
	2.105	0.331	

Table 26: Wilcoxon MRE significance test of KC1 individual, best linear, and best nonlinear (p-level = 0.1)

Figure 40 box plots KC1 best individual model against best linear and nonlinear ensembles. Both ensembles, has smaller box, and whisker than best individual (i.e. better in performance). However, nonlinear ensemble had the narrowest box, and the smallest whisker (i.e. the best prediction model). Furthermore, Figure 40 supports our hypothesis that ensembles offer competitive, even better performance than individual models. Between ensembles, nonlinear ensembles proved to be the best candidate for building future prediction models, since it provides promising results.

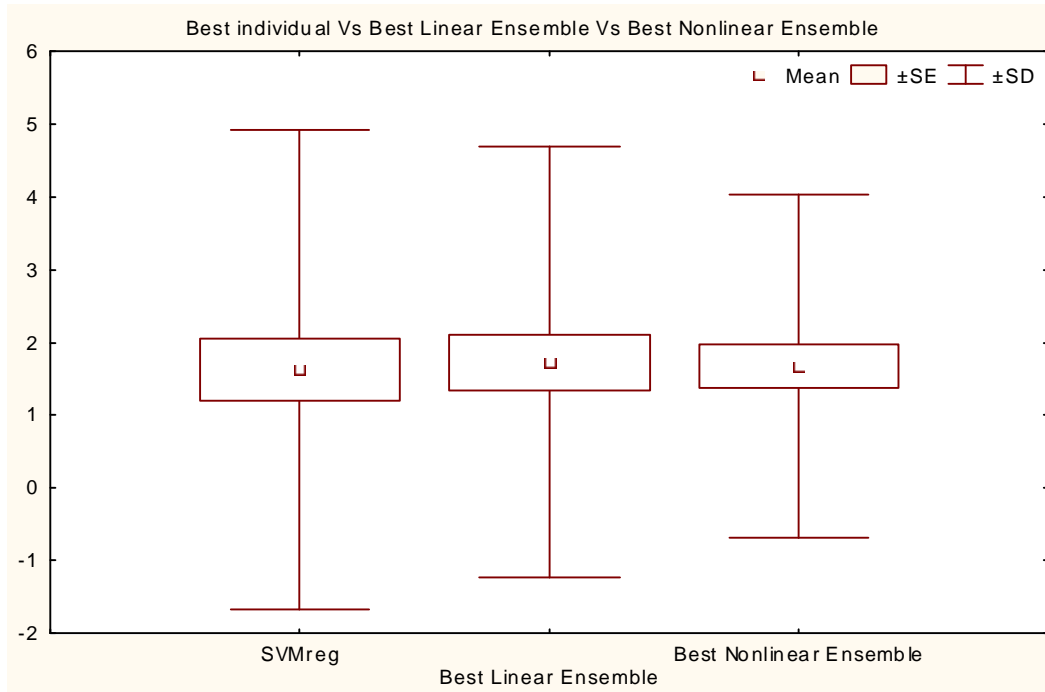


Figure 40: KC1 box plots of best individual Vs best linear Vs best nonlinear (regression)

6.3.2 UIMS

Table 27 compares the results obtained from applying individual regression models on UIMS maintainability dataset. MLP achieved the best MMRE, and StdMRE, while SVMreg was second best with a minor difference in both measures of (0.25), and (0.02), respectively. Also, MLP scored the best Pred(0.3), alongside with M5P. However, RBF was the worst performed regression model in all measures.

	Individual models			
	MLP	RBF	SVMreg	M5P
MMRE	1.39	3.23	1.64	1.67
StdMRE	2.40	4.43	2.38	2.75
Pred(0.3)	23.33	15	20	23.33

Table 27: UIMS results for individual models (regression)

Figure 41 shows the MRE box plots of individual regression models. It is observed that both MLP and SVMreg compete for the narrowest box, and smallest whisker. However, both MLP box and whisker are lower than SVMreg (i.e. MLP better than SVMreg). It is clear that RBF had the biggest box and longest whisker (i.e. worst performed model).

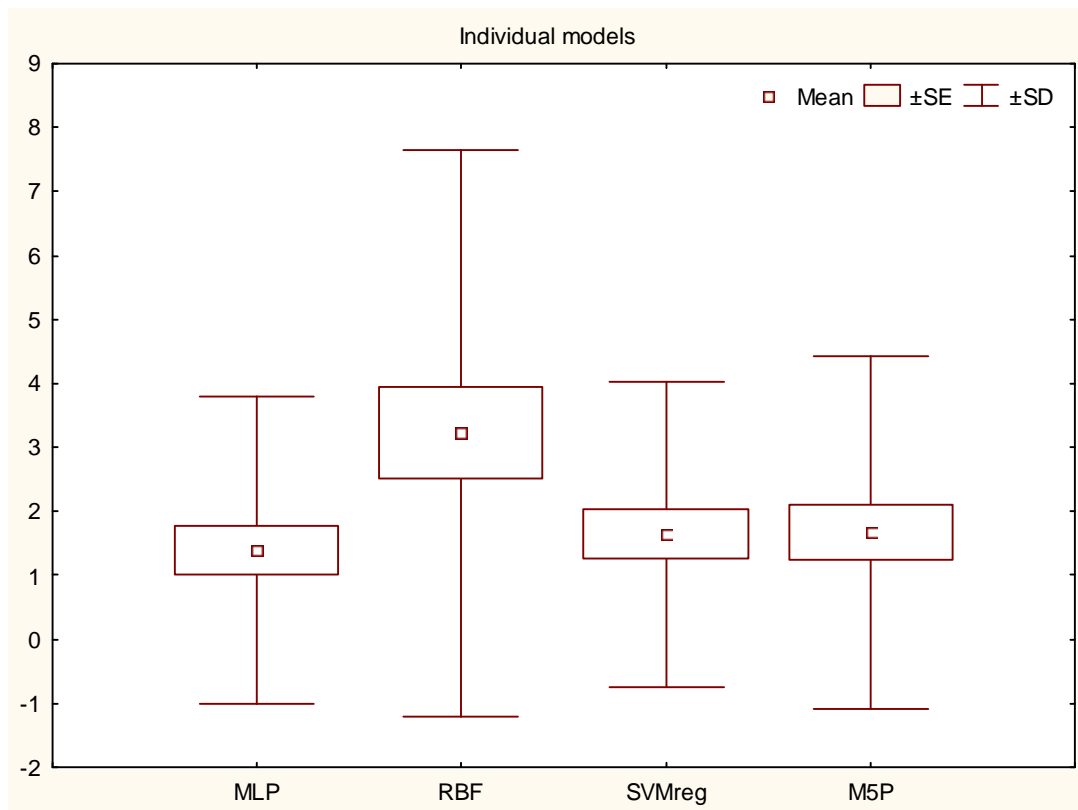


Figure 41: UIMS box plots for individual models (regression)

Table 28 compares the results obtained from different constructed linear ensembles (i.e. average, best, and weight) applied on UIMS maintainability dataset. Best ensemble achieved the lowest MMRE and StdMRE, and highest Pred(0.3). This is due to the reason that different prediction models were chosen for different folds, and these models produced the lowest error. Therefore, best ensemble shows superior performance over average, and weight ensembles.

	Linear Ensemble		
	Average	Best	Weight
		MMRE	MMRE
MMRE	1.46	0.97	1.21
StdMRE	2.08	1.61	1.78
Pred(0.3)	23.33	25	23.33

Table 28: UIMS results for linear ensembles (regression)

Figure 42 shows the MRE box plots of constructed linear ensemble models. Best ensemble box and whisker is lower than of those average, and weight ensemble. Also, best ensemble had the smallest whisker. However, average ensemble had the largest box, and longest whisker.

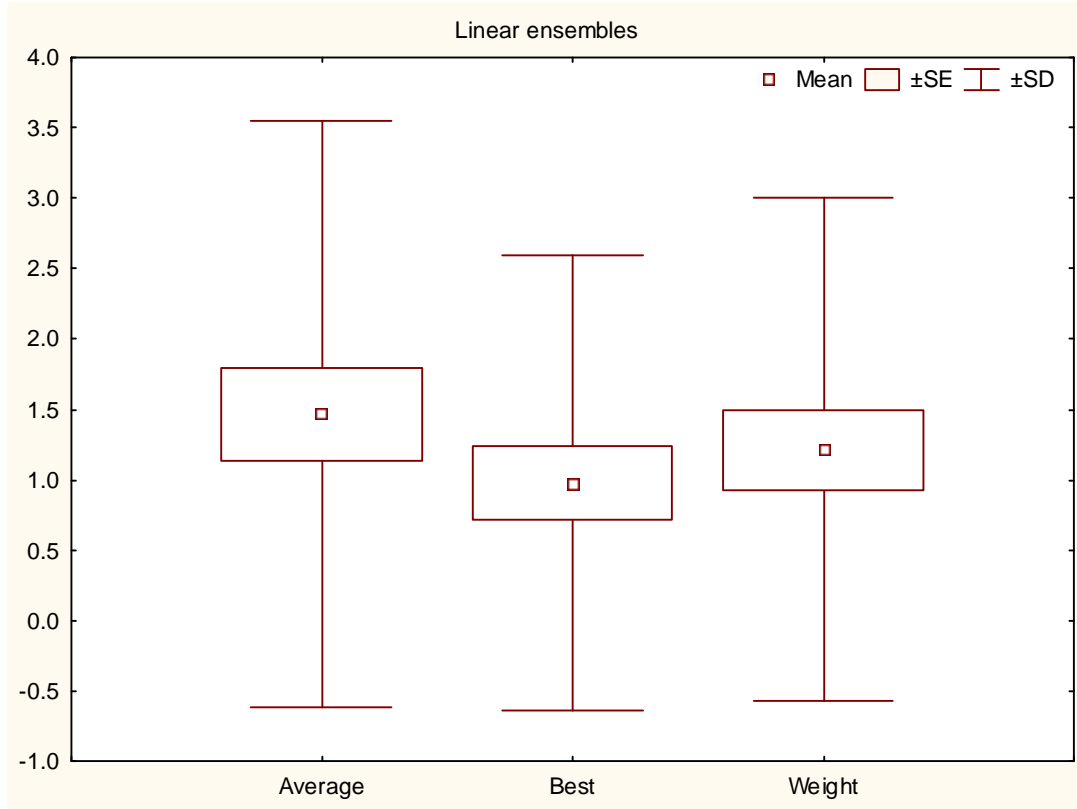


Figure 42: UIMS box plots for linear ensembles (regression)

Table 29 compares the results obtained from nonlinear ensembles constructed using different nonlinear arbitrators (i.e. MLP, RBF, SVMreg, and M5P) applied on UIMS maintainability dataset. SVMreg nonlinear ensemble achieved the best MMRE, and StdMRE, with M5P nonlinear ensemble as second best in both measures with difference of (0.3) and (0.49), respectively. But, M5P nonlinear ensemble scored the highest Pred(0.3) value (i.e. 25%), and SVMreg nonlinear ensemble had the second highest Pred(0.3) value. However, RBF nonlinear ensemble was the worst performed nonlinear ensemble, since it scored the worst values in all evaluation measures.

	Nonlinear Ensemble			
	Nonlinear models			
	MLP	RBF	SVMreg	M5P
MMRE	1.26	3.03	0.93	1.23
StdMRE	2.07	3.98	1.24	1.73
Pred(0.3)	15	13.33	21.66	25

Table 29: UIMS results for nonlinear ensembles (regression)

Figure 43 shows the MRE box plot of different nonlinear ensembles. It is clearly observed that SVMreg nonlinear ensemble outperformed all other nonlinear ensembles, since it had the narrowest box, smallest whisker, and the lowest box and whisker. While, RBF nonlinear ensemble was the worst among nonlinear ensembles (i.e. biggest box, and longest whisker).

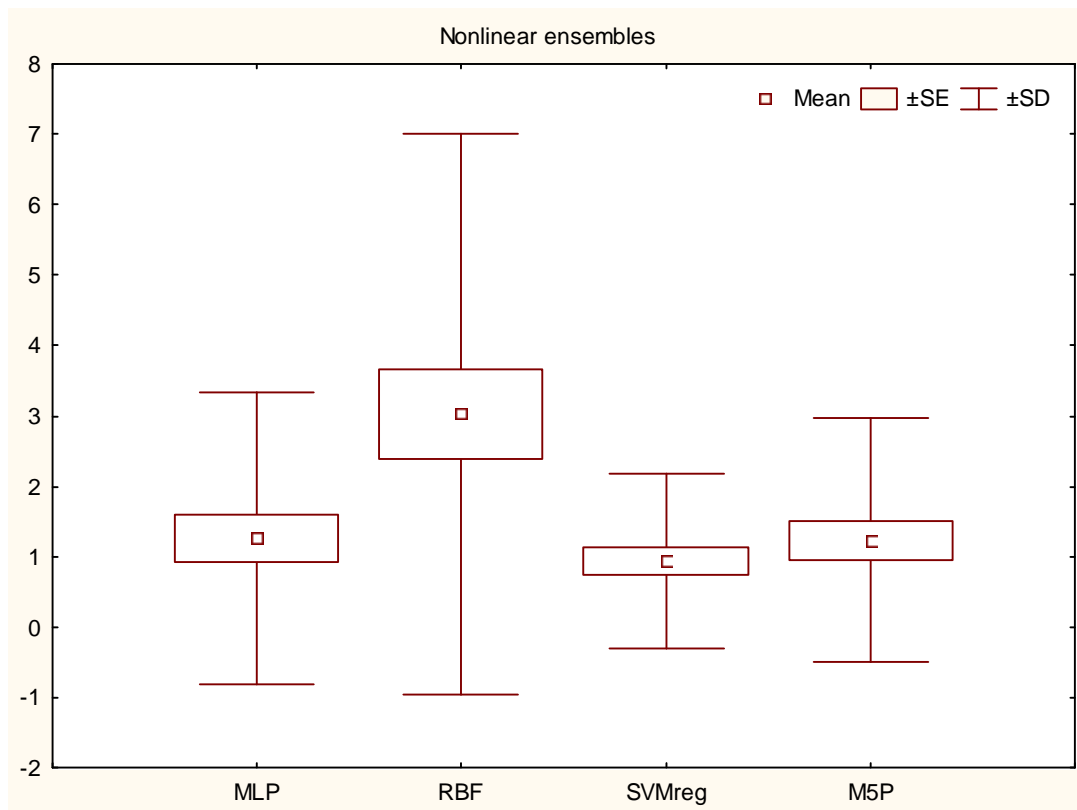


Figure 43: UIMS box plots for nonlinear ensembles (regression)

Now, we discussed the performance of various models in each category (i.e. individual, linear ensemble, nonlinear ensemble models). Next, we pick the best model from each category as Table 30 shows, and compare them, to examine to which extent ensembles offer an increase in performance. MLP was the best model as an individual model, best ensemble as linear ensemble, and SVMreg ensemble as nonlinear ensemble.

As Table 30 shows, SVMreg nonlinear ensemble achieved the best MMRE (i.e. 0.93), and StdMRE (i.e. 1.24), with best linear ensemble as second best. However, best linear ensemble scored the highest Pred(0.3) value (i.e. 25%). Both linear and nonlinear ensembles offer a considerable amount of increase in performance over individual models.

Moreover, Table 31 supports our findings, since the difference between SVMreg nonlinear ensemble and MLP is significant. However, the difference between linear and nonlinear ensembles was not significant.

	Individual	Linear	Nonlinear
		Best	
	MLP	MMRE	SVMreg
MMRE	1.39	0.97	0.93
StdMRE	2.40	1.61	1.24
Pred(0.3)	23.33	25	21.66

Table 30: UIMS comparison of individual Vs best linear Vs best nonlinear (regression)

	MLP	Weight linear ensemble	SVMreg nonlinear ensemble
MLP			
Weight linear ensemble	0.376		
	0.885		
SVMreg nonlinear ensemble	0.074	0.209	
	1.786	1.256	

Table 31: Wilcoxon MRE significance test of UIMS individual, best linear, and best nonlinear (p-level = 0.1)

Figure 44 shows the MRE box plot of best individual regression model against linear and nonlinear ensembles. It is observed that both ensembles outperformed best individual model, since they have smaller boxes and whiskers, and lower than those of MLP (i.e. best individual model). In addition, between ensembles, nonlinear ensemble had the smallest box and whisker, and lower than those of linear ensemble.

In summary, ensembles in general offer competitive, in fact, better performance than individual models, and nonlinear ensembles proved to be the best among ensembles, since it provides promising results in the field of software maintainability prediction.

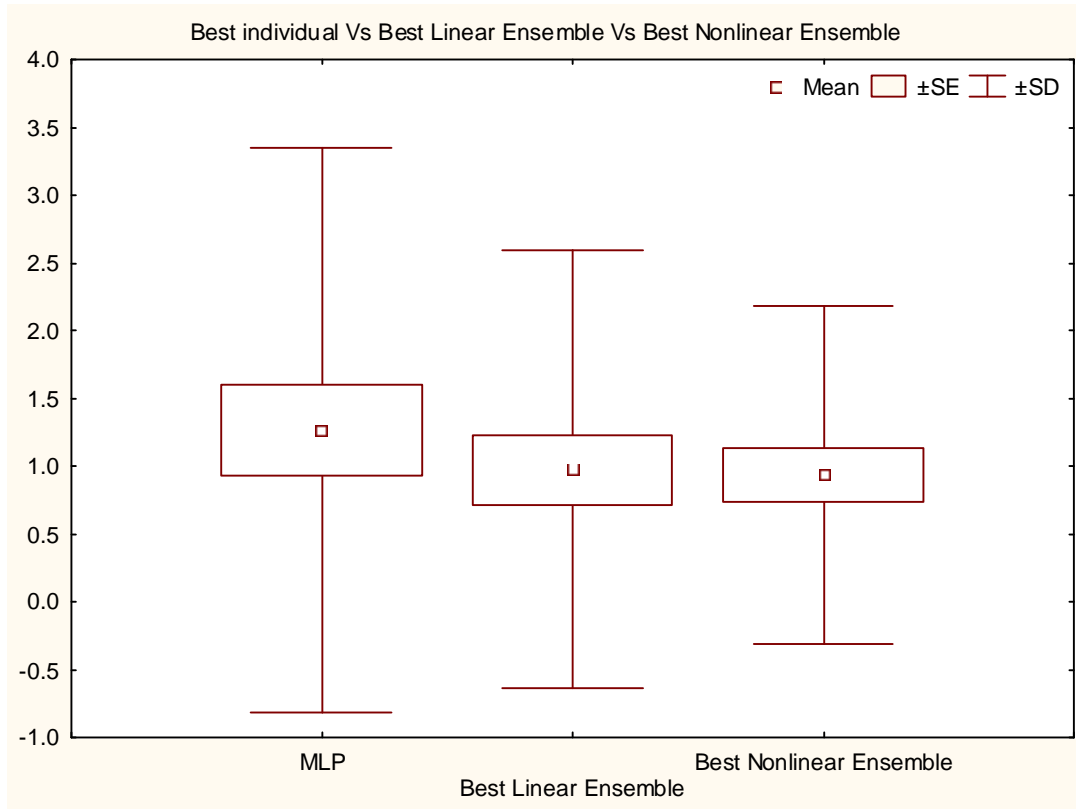


Figure 44: UIMS box plots of best individual Vs best linear Vs best nonlinear (regression)

6.3.3 QUES

Table 32 compares the results obtained from applying individual regression models on QUES maintainability dataset. SVMreg outperformed MLP, RBF, and M5P in all evaluation measures. SVMreg achieved the best value in both MMRE and StdMRE (i.e. lowest MMRE value). Moreover, SVMreg scored the highest Pred(0.3) value (i.e. 56.6%). However, M5P showed a competitive result compared with SVMreg, it was achieved the second best in all evaluation measures. RBF was the worst performed model (i.e. highest MMRE and StdMRE, and lowest Pred(0.3) value).

	Individual model			
	MLP	RBF	SVMreg	M5P
MMRE	0.71	0.96	0.44	0.54
StdMRE	0.65	1.52	0.39	0.56
Pred(0.3)	40	36.66	56.66	51.66

Table 32: QUES results for individual models (regression)

Figure 45 shows the MRE box plots of individual regression models. It clearly observed that SVMreg outperformed other regression models, since it had the narrowest box, smallest whisker, and the level of its box and whisker is lower than those of other regression models. In the other hand, RBF had the biggest box, longest whisker (i.e. worst model).

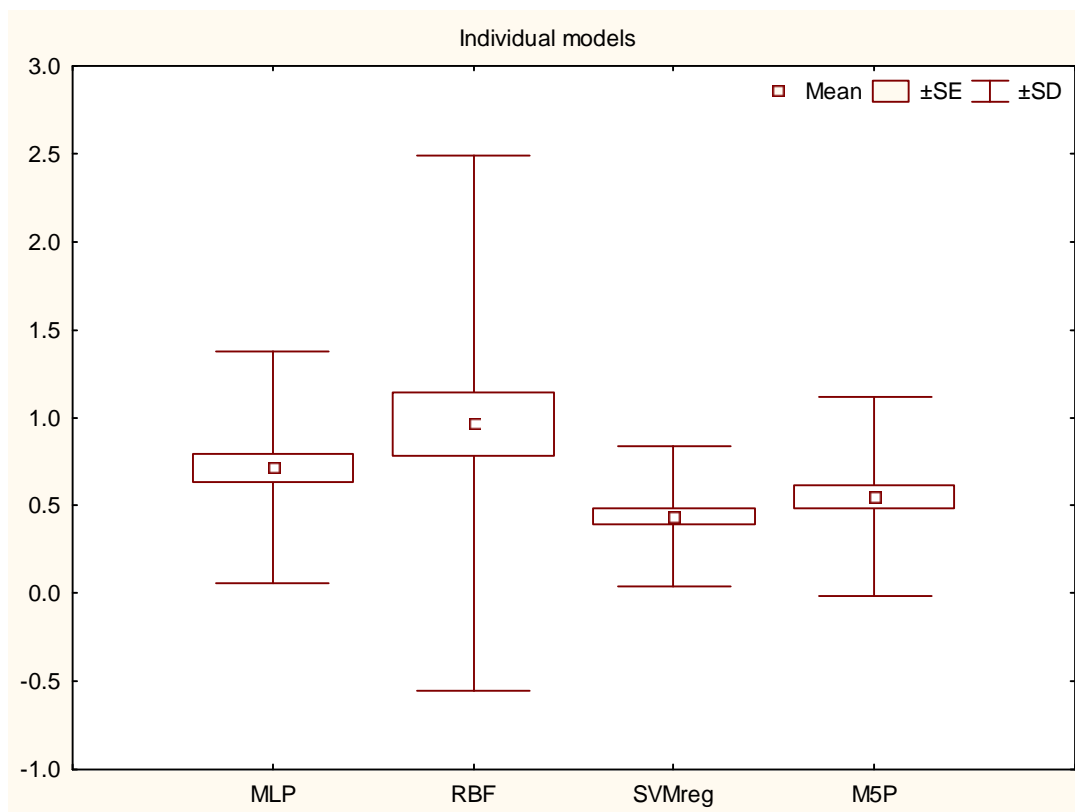


Figure 45: QUES box plots for individual models (regression)

Table 33 compares the results obtained from different constructed linear ensembles (i.e. average, best, and weight) applied on QUES maintainability dataset. All created linear ensembles shows competitive results, however, best ensemble outperformed all other linear ensembles in all evaluation measures. Best ensemble achieved the lowest MMRE and StdMRE (i.e. 0.4 and 0.3, respectively), and highest Pred(0.3) value (i.e. 60%).

	Linear Ensemble		
	Average	Best	Weight
		MMRE	MMRE
MMRE	0.58	0.41	0.49
StdMRE	0.69	0.32	0.51
Pred(0.3)	53.33	60	53.33

Table 33: QUES results for linear ensembles (regression)

Figure 46 shows the MRE box plots of constructed linear ensembles. It is clearly observed that best ensemble outperformed all other linear ensembles. Best ensemble had the narrowest box, and smallest whisker. Furthermore, the level of its box and whisker is lower than those of average and weight ensemble. However, average ensemble was the worst linear ensemble (i.e. biggest box and longest whisker).

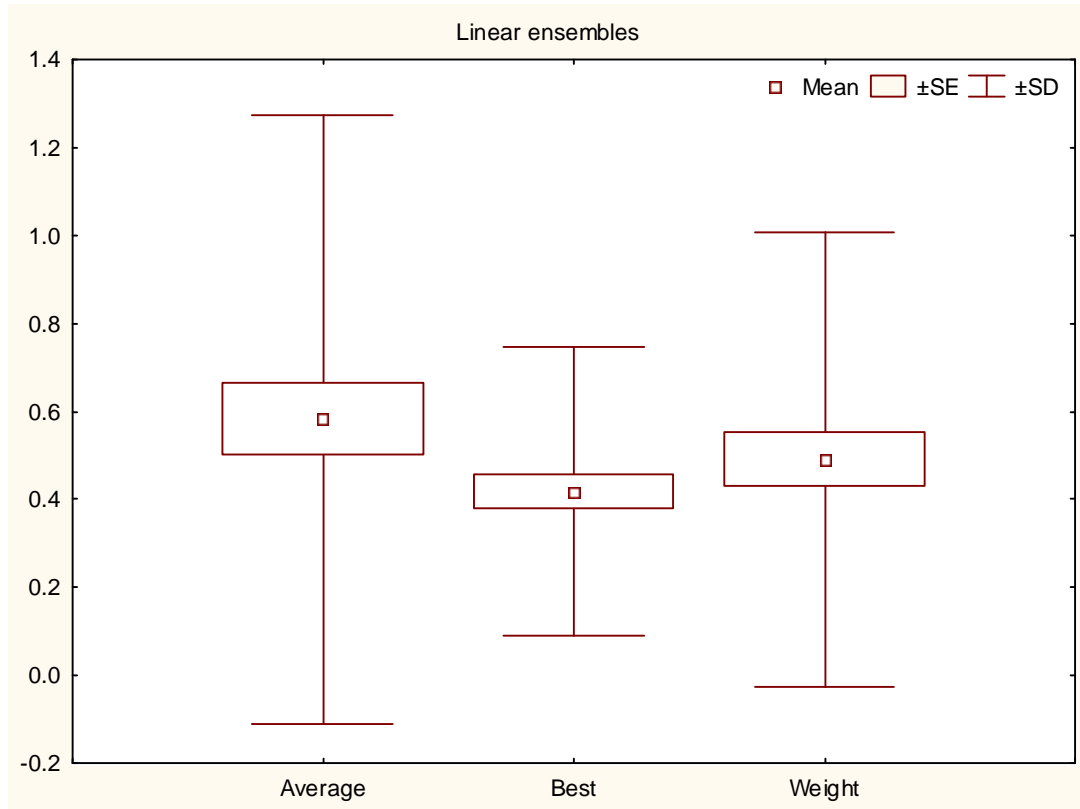


Figure 46: QUES box plots for linear ensembles (regression)

Table 34 compares the results obtained from nonlinear ensembles constructed using different nonlinear arbitrators (i.e. MLP, RBF, SVMreg, and M5P) applied on QUES maintainability dataset. SVMreg and M5P nonlinear ensembles are competing for the best nonlinear ensemble. However, SVMreg ensemble was slightly better than M5P ensemble in all evaluation measures with difference of (0.06), (0.04), and (1.7%) in MMRE, StdMRE, and Pred(0.3), respectively. Therefore, it can be concluded that SVMreg ensemble is the best nonlinear ensemble. However, RBF ensemble was the worst performed model (i.e. highest MMRE and StdMRE, and lowest Pred(0.3) value).

	Nonlinear Ensemble			
	Nonlinear models			
	MLP	RBF	SVMreg	M5P
MMRE	0.57	0.92	0.38	0.44
StdMRE	0.54	1.44	0.35	0.39
Pred(0.3)	50	41.66	60	58.33

Table 34: QUES results for nonlinear ensembles (regression)

Figure 47 shows the MRE box plots of constructed nonlinear ensembles. Both SVMreg and M5P ensembles is competing for the smallest box. However, it can be clearly observed that SVMreg ensemble had smaller whisker than other nonlinear ensembles, and the level of its box and whisker is lower than those of MLP, RBF, and M5P ensembles. RBF ensemble was the worst linear ensemble (i.e. biggest box and longest whisker).

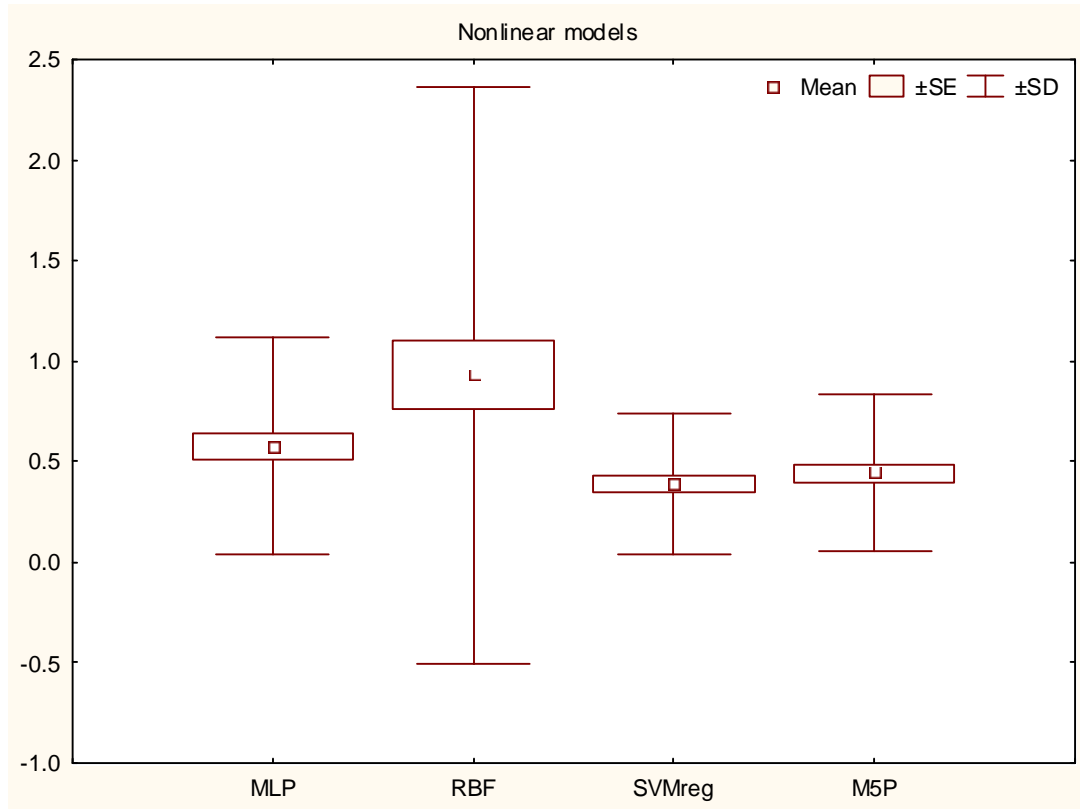


Figure 47: QUES box plots for nonlinear ensembles (regression)

Now, we discussed the performance of various models in each category (i.e. individual, linear ensemble, nonlinear ensemble models). Next, we pick the best model from each category as Table 35 shows, and compare them, to examine to which extent ensembles offer an increase in performance. SVMreg was the best model as an individual model, best ensemble as linear ensemble, and SVMreg ensemble as nonlinear ensemble.

As Table 35, nonlinear ensemble achieved the best MMRE (i.e. 0.38), with linear ensemble as second best MMRE achieved (i.e. 0.41). In the other hand, linear ensemble had the best StdMRE (i.e. 0.32), with nonlinear ensemble as second best StdMRE (i.e. 0.35). In terms of Pred(0.3) , both ensembles (i.e. linear and nonlinear) scored the highest Pred(0.3) value (i.e. 60%). However, the difference between individual model and ensembles was not significant, as indicated by Table 36.

	individual	Linear	Nonlinear
		Best	
	SVMreg	MMRE	SVMreg
MMRE	0.44	0.41	0.38
StdMRE	0.39	0.32	0.35
Pred(0.3)	56.66	60	60

Table 35: QUES comparison of individual Vs best linear Vs best nonlinear (regression)

	MLP	Best linear ensemble	SVMreg nonlinear ensemble
MLP			
Best linear ensemble	0.281 1.079		
SVMreg nonlinear ensemble	0.122 1.547	0.289 1.060	

Table 36: Wilcoxon MRE significance test of QUES individual, best linear, and best nonlinear (p-level = 0.1)

Figure 48 shows the MRE box plot of best individual regression model against linear and nonlinear ensembles. It is observed that both ensembles have smaller whisker than best individual model, and the level of their box and whisker is lower than those of best individual model. Thus, both ensembles offer an increase in performance over best individual model. Between ensembles, nonlinear box is lower than the box of best linear ensemble. However, best linear ensemble has smaller whisker, and slightly smaller box.

In summary of this chapter, ensembles in general offer competitive, or even, better performance than individual models, and they provide promising results in the field of software maintainability prediction.

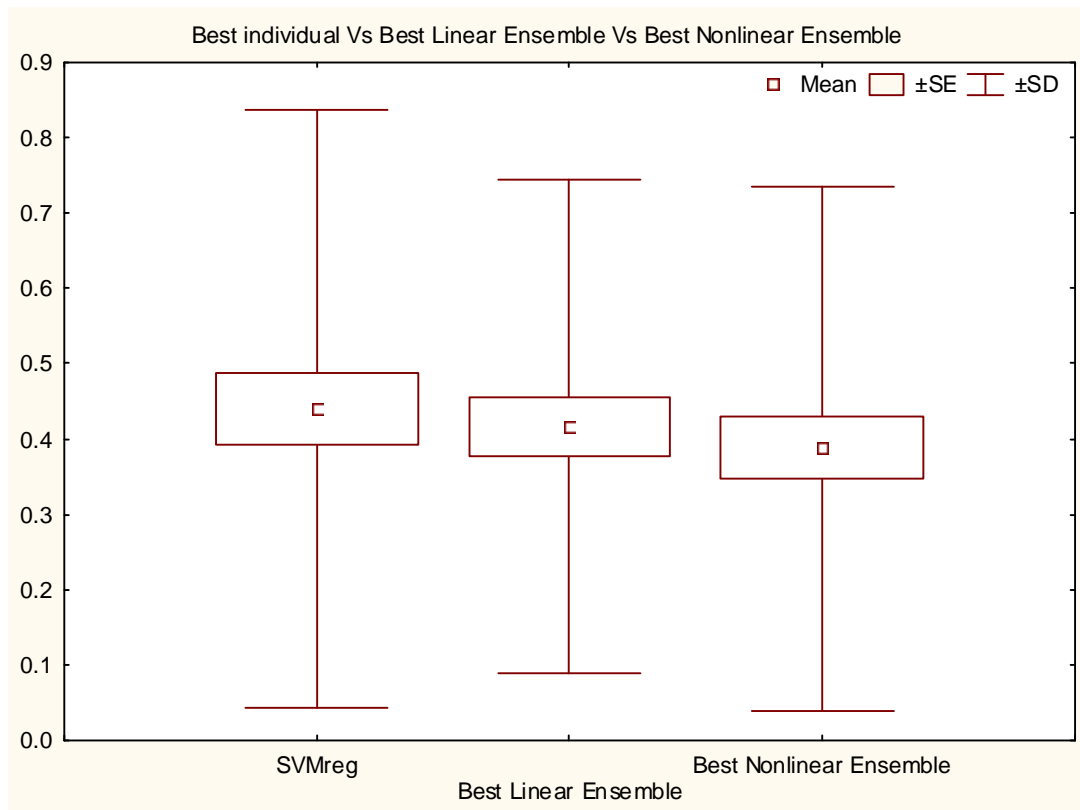


Figure 48: QUES box plots of best individual Vs best linear Vs best nonlinear (regression)

CHAPTER 7

Conclusion

The main objective of this thesis was to build different ensemble models, and evaluate their prediction accuracy against stand-alone prediction models. Several linear and nonlinear ensembles were built, and three empirical studies were conducted to evaluate them in the context of fault and maintenance effort prediction.

Overall empirical results produced in this thesis show that ensembles in general offer better, or at least competitive, performance than individual models. In addition, nonlinear ensembles achieved the best accuracy among ensembles.

The rest of this chapter is organized as follows: thesis contributions are discussed in the next section. After that, the limitations of this thesis are discussed. Finally, directions for future work were provided.

7.1 Thesis contributions

Throughout our empirical studies, we provided empirical evidences and interesting results for both software quality assurance and computational intelligence communities. These contributions are summarized below:

- Investigated six popular and common computational intelligence models (i.e. BBN, NB, MLP, RBF, SVM, DT) in identifying faulty classes. Some of these models, such as SVM and BBN, were not investigated before for the same purpose.

- Results indicate that MLP was superior over other individual models.
- Evaluated the prediction capability of single model ensembles (i.e. bagging and boosting) over individual models, in the context of identifying faulty classes in a software system.
 - Empirical results indicate that bagging and boosting yield improved classification accuracy over most of the investigated single classifiers. However, bagging and boosting performance varied from one classifier to another. In some cases, bagging outperforms boosting, while in some other cases, boosting outperforms bagging.
 - In case of MLP and NB, bagging produced the best accuracy. In case of RBF and DT, boosting produced the best accuracy. However, in case of SVM, bagging and boosting resulted in detrimental in accuracy.
- Proposed different multi-model linear ensembles for classification domain (i.e. majority voting, average probability, best probability, and weighted probability ensembles).
- Evaluated the proposed multi-model linear ensembles (classification) against individual prediction models, in the context of identifying faulty classes in a software system.
 - Empirical results indicate that multi-model linear ensembles show competitive results in classification accuracy.
- Proposed different nonlinear arbitrators to build different multi-model nonlinear ensembles for classification domain

- Empirical studies indicate that nonlinear ensembles are promising models for providing improved accuracy.
- Proposed different multi-model linear ensembles for regression domain (i.e. average, best, and weight ensembles).
- Evaluated the proposed multi-model linear ensembles (regression) against individual prediction models, in the context of estimating the fault density of faulty classes, and maintenance effort.
 - Experimental results indicate that multi-model linear ensembles in general offer better, or at least competitive, performance.
- Proposed different nonlinear arbitrators to build different multi-model nonlinear ensembles for regression domain.
 - Empirical studies indicate that nonlinear ensembles are promising models for providing improved accuracy.

7.2 Limitations

We will list limitations faced in this research, along with our point of view for these limitations:

- Choice of individual models: we chose six individual models for classification domain, while only four out of these six were applicable in the regression domain. However, we selected these models across different categories to achieve a balance between established prediction models, and we selected the models that are commonly and widely used in the literature of software quality prediction.
- Parameter initialization: in our experiments, we set the models parameters, to the default settings in WEKA, without any parameter optimization. However, our

goal was to build different ensemble models from stand-alone models, without taking into consideration the optimization issue. In addition, it has been observed that the default settings were used in many papers in the literature. Further studies are needed to further support findings of this research.

- Generalization: we used one dataset for class fault classification, one dataset for class fault density estimation, and two datasets for maintenance effort prediction. More studies are needed to further support the findings of this research.
- Some models perform well in training; however, in testing they may perform the worst. This will decrease the accuracy of linear ensembles performance, since, these models relay on training performance for their construction (e.g. weight and best probability ensemble in classification domain).

7.3 Future work

The field of software quality prediction is an interesting field, which provides valuable information for both software quality assurance and computational intelligence communities. After conducting this research, we thought of a number of suggestions to extend the directions of this research. Such suggestions include:

- Consider other stand-alone prediction models to construct different ensembles.
- Apply ensemble models to other software quality prediction problems.
- Try parameter optimization, before ensemble building.
- Propose and evaluate other weight mechanisms for building weight ensembles.
- In weight and best ensembles, try different error criterion for ranking (e.g. Pred(0.3) for regression and F-measure for classification).

References

- [1] "NASA IV&V Facility Metrics Data Program," <http://mdp.ivv.nasa.gov/index.html>.
- [2] A. Abraham, "Artificial Neural Networks."
- [3] M. A. D. Almeida and S. Matwin, "Machine Learning Method for Software Quality Model Building," in *Proceedings of the 11th International Symposium on Foundations of Intelligent Systems*: Springer-Verlag, 1999.
- [4] N. Baba, "A new approach for finding the global minimum of error function of neural networks," *Neural Network*, vol. 2, pp. 367-373, 1989.
- [5] R. K. Bandi, V. K. Vaishnavi, and D. E. Turk, "Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics," *IEEE Transactions on Software Engineering*, vol. 29, pp. 77-87, 2003.
- [6] R. Banfield, L. Hall, K. Bowyer, and W. Kegelmeyer, "A Comparison of Decision Tree Ensemble Creation Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, 2007.
- [7] V. R. Basili, L. C. Briand, and W. L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, vol. 22, pp. 751 - 761, 1996.
- [8] B. Bhattacharya and D. P. Solomatine, "Machine learning in sedimentation modelling," *Neural Netw.*, vol. 19, pp. 208-214, 2006.
- [9] J. M. Bieman and B.-K. Kang, "Cohesion and Reuse in an Object-Oriented System," in *ACM Symposium on Software Reusability (SSR'94)*, 1995, pp. 259-262.

- [10] V. G. Bittencourt, M. C. C. Abreu, M. C. P. d. Souto, and A. M. d. P. Canuto, "An empirical comparison of individual machine learning techniques and ensemble approaches in protein structural class prediction," in *International Joint Conference on Neural Networks*, 2005, pp. 527 - 531.
- [11] B. W. Boehm, *Software Engineering Economics*: Prentice Hall PTR, 1981.
- [12] B. W. Boehm and P. N. Papaccio, "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering*, vol. 14, pp. 1462-1477, 1988.
- [13] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145-1159, 1997.
- [14] P. L. Braga, A. L. I. Oliveira, G. H. T. Ribeiro, and S. R. L. Meira, "Bagging Predictors for Estimation of Software Project Effort," in *International Joint Conference on Neural Networks*, 2007, pp. 1595 - 1600.
- [15] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123-140, 1996.
- [16] L. Briand, P. Devanbu, and W. Melo, "An Investigation into Coupling Measures for C++," in *Proceedings of the 19th international conference on Software engineering (ICSE '97)*, Boston, USA, 1997, pp. 412-421.
- [17] L. C. Briand, C. Bunse, and J. W. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs," *IEEE Transactions on Software Engineering*, vol. 27, pp. 513-530, 2001.
- [18] L. C. Briand, W. L. Melo, and J. Wüst, "Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects " *IEEE Transactions on Software Engineering*, vol. 28, pp. 706 - 720, 2002.

- [19] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems," *Journal of Systems and Software*, vol. 51, pp. 245 - 273, 2000.
- [20] M. Cartwright and M. Shepperd, "An Empirical Investigation of an Object-Oriented Software System," *IEEE Transactions on Software Engineering*, vol. 26, pp. 786 - 796, 2000.
- [21] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 302–309, 1991.
- [22] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, pp. 476 - 493, 1994.
- [23] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*: Benjamin-Cummings Publishing Co., Inc., 1986.
- [24] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, pp. 273-297, 1995.
- [25] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.
- [26] B. Cukic, "Guest Editor's Introduction: The Promise of Public Software Engineering Data Repositories," *IEEE Software*, vol. 22, pp. 20 - 22, 2005.
- [27] J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," in *23rd international conference on Machine learnin*, 2006, pp. 233-240.

- [28] K. El-Emam, W. Melo, and J. C. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," *Journal of Systems and Software*, vol. 56, pp. 63 - 75, 2001.
- [29] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of Systems and Software*, vol. 81, pp. 649-660, 2008.
- [30] M. O. Elish and K. O. Elish, "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study," in *13th European Conference on Software Maintenance and Reengineering (CSMR '09)* 2009, pp. 69 - 78.
- [31] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861-874, 2006.
- [32] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*: PWS Publishing Co., 1998.
- [33] F. Fioravanti and P. Nesi, "Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 27, pp. 1062-1084, 2001.
- [34] F. Fioravanti and P. Nesi, "A study on fault-proneness detection of object-oriented systems," in *Fifth European Conference on Software Maintenance and Reengineering*, 2001, pp. 121-130.
- [35] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," *IEEE Transactions on Software Engineering*, vol. 29, pp. 985-995, 2003.
- [36] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, pp. 256-285, 1995.

- [37] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, 1995, pp. 23-37.
- [38] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *In Proceedings of the Thirteenth International Conference on Machine Learning*, Italy, 1996, pp. 148-156.
- [39] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Thirteenth International Conference on Machine Learning*, Italy, 1996, pp. 148-156.
- [40] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, pp. 131-163, 1997.
- [41] D. Galin, *Software Quality Assurance, From Theory to Implementation*, 1st ed.: Pearson, Addison-Wesley, 2004.
- [42] D. Grossman and P. Domingos, "Learning Bayesian network classifiers by maximizing conditional likelihood," in *Proceedings of the twenty-first international conference on Machine learning* Banff, Alberta, Canada: ACM, 2004.
- [43] S. Gutta and H. Wechsler, "Face Recognition Using Hybrid Classifier Systems," in *IEEE International Conference on Neural Networks*, 1996, pp. 1017-1022.
- [44] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Transactions on Software Engineering*, vol. 31, pp. 897 - 910, 2005.
- [45] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural network design*: PWS Publishing Co., 1996.

- [46] L. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993-1001, 1990.
- [47] W. Harrison, "Using software metrics to allocate testing resources," *Journal of Management Information Systems*, vol. 4, pp. 93-105, 1988.
- [48] S. Hashem, B. Schmeiser, and Y. Yih, "Optimal linear combinations of neural networks." vol. 3: Neural Networks, 1994, pp. 1507-1512.
- [49] S. Haykin, *Neural Networks: A Comprehensive Foundation* New Jersey, USA: Prentice Hall, 1999.
- [50] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems," in *International Symposium on Applied Corporate Computing* Mexico, 1995.
- [51] F. J. Huang, Z. Zhou, H.-J. Zhang, and T. Chen, "Pose invariant face recognition," in *In Proc. 4th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, France, 2000, pp. 245-250.
- [52] IEEE, "IEEE standard glossary of software engineering terminology, report IEEE Std 610.12-1990," *IEEE*, 1990.
- [53] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *IEEE Computer Magazine*, vol. 29, pp. 31-44, 1996.
- [54] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, Second ed.: Springer Verlag 2007.
- [55] T. Kamiya, S. Kusumoto, and K. Inoue, "Prediction of fault-proneness at early phase in object-oriented development," in *Second IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, 1999, pp. 253 – 258.

- [56] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Information and Software Technology*, vol. 49, pp. 483-492, 2007.
- [57] T. M. Khoshgoftaar, E. Geleyn, and L. Nguyen, "Empirical Case Studies of Combining Software Quality Classification Models," in *Third International Conference on Quality Software*, 2003, p. 40.
- [58] N. R. Kiran and V. Ravi, "Software reliability prediction by soft computing techniques," *Journal of Systems and Software*, vol. 81, pp. 576-583, 2008.
- [59] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure," *IEEE Software*, vol. 148, pp. 81-85, 2001.
- [60] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, pp. 1137-1143.
- [61] A. G. Koru and H. Liu, "Building Defect Prediction Models in Practice," *IEEE Software*, vol. 22, pp. 23-29, 2005.
- [62] C. v. Kotten and A. R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability," *Information and Software Technology*, vol. 48, pp. 59 - 67, 2006.
- [63] A. Krogh and J. Vedelsby, "Neural Network Ensembles, Cross Validation, and Active Learning," *Advances in Neural Information Processing Systems*, vol. 7, pp. 231-238, 1995.
- [64] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *National Conference on Artificial Intelligence* 1992, pp. 223-228
- [65] Y. Lee, B. Liang, S. Wu, and F. Wang, "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow," in *International Conference on Software Quality Slovenia*, 1995.

- [66] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems and Software*, vol. 23, pp. 111-122, 1993.
- [67] A. D. Lucia, E. Pompella, and S. Stefanucci, "Assessing effort estimation models for corrective maintenance through empirical studies," *Information and Software Technology*, vol. 47, pp. 3 - 15, 2005.
- [68] S. G. MacDonell, "Establishing relationships between specification size and software process effort in case environment," *Information and Software Technology*, vol. 39, pp. 35-45, 1997.
- [69] A. Mahaweerawat, P. Sophatsathit, C. Lursinsap, and P. Musilek, "Fault Prediction in Object-Oriented Software Using Neural Network Techniques," in *International Conference on Intelligent Technologies*, 2004, pp. 27-34.
- [70] J. Mao, "A case study on bagging, boosting and basic ensembles of neural networks for OCR," in *In Proc. IEEE Int. Joint Conf. on Neural Networks*, 1998, pp. 1828-1833.
- [71] J. Mingers, "An Empirical Comparison of Pruning Methods for Decision Tree Induction," *Machine Learning*, vol. 4, pp. 227-243, 1989.
- [72] S. C. Misra, "Modeling Design/Coding Factors That Drive Maintainability of Software Systems," *Software Quality Control*, vol. 13, pp. 297-320, 2005.
- [73] J. D. Musa, *Software Reliability Engineering: More Reliable Software Faster and Cheaper*. Authorhouse, 2004.
- [74] I. Myrtveit and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Transactions on Software Engineering*, vol. 31, pp. 380 - 391, 2005.
- [75] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169-198, 1999.

- [76] D. W. Opitz and R. F. Maclin, "An empirical evaluation of bagging and boosting for artificial neural networks," in *International Joint Conference on Neural Networks*, 1997, pp. 1401-1405.
- [77] D. W. Opitz and J. W. Shavlik, "Actively searching for an effective neural-network ensemble," *Connection Science*, vol. 8, pp. 337-353, 1996.
- [78] D. W. Opitz and J. W. Shavlik, "Generating Accurate and Diverse Members of a Neural-Network Ensemble," *Advances in Neural Information Processing Systems*, vol. 8, pp. 535--541, 1996.
- [79] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.
- [80] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481-1497, 1990.
- [81] A. A. Porter and R. W. Selby, "Empirically Guided Software Development Using Metric-Based Classification Trees," *IEEE Software*, vol. 7, pp. 46-54, 1990.
- [82] R. S. Pressman, *Software engineering: a practitioner's approach (Sixth ed.)*: McGraw-Hill, Inc., 1986.
- [83] J. R. Quinlan, "Bagging, boosting, and c4.5," in *Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 725-730.
- [84] J. R. Quinlan, *C4.5: Programs for Machine Learning*: Morgan Kaufmann Publishers, 1993.
- [85] R. J. Quinlan, "Learning with Continuous Classes," in *5th Australian Joint Conference on Artificial Intelligence*, Singapore, 1992, pp. 343-348.
- [86] I. Rish, "An empirical study of the naive Bayes classifier," in *Workshop on Empirical Methods in Artificial Intelligence (IJCAI 2001)*, 2001.

- [87] S. R. Safavian and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology," *IEEE transactions on Systems, Man and Cybernetics*, vol. 21, pp. 660-674, 1991.
- [88] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197-227, 1990.
- [89] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to the SMO Algorithm for SVM Regression," *IEEE Transactions on neural networks*, vol. 11, pp. 1188-1193, 2000.
- [90] Y. Shimshoni and N. Intrator, "Classification of seismic signals by integrating ensembles of neural networks," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1194-1201, 1998.
- [91] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199-222, 2004.
- [92] P. Sollich, "Learning with Ensembles: How over-fitting can be useful," *Advances in Neural Information Processing Systems*, vol. 8, pp. 190-196, 1996.
- [93] R. Subramanyam and M. S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," *IEEE Transactions on Software Engineering*, vol. 29, pp. 297 - 310, 2003.
- [94] M.-H. Tang, M.-H. Kao, and M.-H. Chen, "An empirical study on object-oriented metrics," in *6th International Symposium on Software Metrics*, 1999, p. 242.
- [95] M. M. T. Thwin and T.-S. Quah, "Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics," *Journal of Systems and Software*, vol. 76, pp. 147 - 156, 2005.
- [96] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer, 1995.

- [97] V. Vapnik, S. E. Golowich, and A. Smola, "Function Approximation, Regression Estimation, and Signal Processing," in *Advances in Neural Information Processing Systems 9*: MIT Press, 1997, pp. 281–287.
- [98] L. Virine and M. Trumper, *Project Decisions: The Art and Science* Management Concepts, 2008.
- [99] S.-j. Wang, A. Mathew, Y. Chen, L.-f. Xi, L. Ma, and J. Lee, "Empirical analysis of support vector machine ensemble classifiers," 2008.
- [100] Y. Wang and I. H. Witten, "Induction of model trees for predicting continuous classes," in *Poster papers of the 9th European Conference on Machine Learning*, 1997.
- [101] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, second ed. San Francisco: Morgan Kaufmann, 2005.
- [102] H. Yan, Y. Jiang, J. Zheng, C. Peng, and Q. Li, "A multilayer perceptron-based medical decision support system for heart disease diagnosis," *Expert Systems with Applications*, vol. 30, pp. 272–281, 2003.
- [103] L. Yu, K. K. Lai, and S. Wang, "Multistage RBF neural network ensemble learning for exchange rates forecasting," *Neurocomputing*, vol. 71, pp. 3295–3302, 2008.
- [104] P. Yu, T. Systä, and H. A. Müller, "Predicting Fault-Proneness Using OO Metrics: An Industrial Case Study," in *6th European Conference on Software Maintenance and Reengineering*, 2002, pp. 99 - 107.
- [105] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems* vol. 69, pp. 125–139, 1995.
- [106] C.-X. Zhang, J.-S. Zhang, and G.-Y. Zhang, "An efficient modified boosting method for solving classification problems," *Journal of Computational and Applied Mathematics*, 2007.

- [107] J. Zheng, "Predicting software reliability with neural network ensembles," *Expert Systems with Applications*, 2007.
- [108] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," *Journal of Systems and Software*, vol. 80, pp. 1349-1361, 2007.
- [109] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, pp. 239-263, 2002.

Appendix A – Detailed experiment results

KC1 (classification) results

	BBN		NB		MLP		RBF		SVM		DT	
Actual	Predicted	Probability	Predicted	Probability	Predicted	Probability	Predicted	Probability	Predicted	Probability	Predicted	Probability
T	F	0.914	T	0.003	T	0	T	0.305	F	0.619	T	0
T	F	0.914	T	0.004	T	0.055	T	0.304	F	0.619	T	0
T	T	0.42	T	0.064	T	0.078	T	0.37	F	0.619	T	0.125
T	F	0.914	T	0.019	T	0.053	T	0.308	F	0.619	T	0
T	T	0.42	T	0.11	T	0	T	0.338	F	0.619	T	0
T	T	0.42	T	0.448	T	0.202	F	0.528	F	0.747	T	0.125
T	T	0.42	T	0.007	T	0.055	T	0.304	F	0.619	F	0.727
T	T	0.42	T	0.274	T	0.012	T	0.386	F	0.618	T	0
T	F	0.914	F	0.984	T	0.089	F	0.936	F	0.619	F	0.857
T	F	0.914	F	0.984	T	0.089	F	0.936	F	0.619	F	0.857
T	T	0.42	F	1	F	0.992	F	0.939	F	0.612	F	0.857
F	T	0.42	F	0.628	F	0.746	F	0.633	F	0.619	F	1
T	T	0.42	F	0.774	T	0.172	T	0.345	F	0.615	T	0.3
F	T	0.42	F	1	F	1	F	0.936	F	0.62	F	0.857
F	T	0.471	F	0.876	T	0.146	F	0.575	T	0.2	T	0.414
T	T	0.471	T	0.001	T	0.001	T	0.081	F	0.588	T	0.056
T	T	0.471	T	0.048	T	0.008	T	0.097	F	0.588	T	0.056
F	T	0.471	F	0.511	F	0.647	F	0.724	F	0.588	F	1
F	F	0.984	F	0.876	F	0.98	F	0.811	F	0.588	F	1
F	T	0.471	F	0.79	F	0.79	F	0.618	F	0.642	T	0.414
T	T	0.471	F	0.846	T	0.186	F	0.541	F	0.598	T	0.414
F	F	0.984	F	0.962	F	1	F	0.739	F	0.787	F	1
F	F	0.984	F	0.962	F	1	F	0.739	F	0.787	F	1
F	T	0.471	F	0.505	T	0.055	F	0.726	F	0.588	T	0.414
T	T	0.471	T	0.372	T	0.01	F	0.601	T	0.357	T	0.056
T	T	0.471	T	0.398	T	0.015	F	0.722	F	0.588	T	0.414

T	T	0.471	T	0.408	T	0.02	F	0.681	F	0.588	T	0.414
T	T	0.471	F	0.972	F	0.995	F	0.837	F	0.614	T	0.414
T	T	0.462	F	0.507	F	0.827	F	0.736	F	0.584	T	0
F	F	0.983	F	0.821	F	0.847	F	0.973	F	0.584	F	1
F	F	0.983	F	0.93	F	0.788	F	0.927	F	0.697	F	1
T	T	0.462	F	0.863	F	1	F	0.949	F	0.583	F	0.846
T	T	0.462	F	0.91	F	0.999	F	0.861	F	0.584	F	1
F	T	0.462	F	0.827	F	0.983	F	0.867	F	0.533	F	1
F	T	0.462	F	0.829	T	0.302	F	0.975	F	0.548	F	0.846
F	F	0.983	F	0.968	F	1	F	0.95	F	0.692	F	1
F	T	0.462	F	0.887	F	1	F	0.793	F	0.587	F	0.8
F	F	0.983	F	0.954	F	1	F	0.969	F	0.584	F	1
F	F	0.983	F	0.9	F	0.997	F	0.889	F	0.796	F	1
T	T	0.462	T	0.236	T	0.018	F	0.597	F	0.584	T	0.167
F	T	0.462	F	0.753	T	0.415	F	0.975	F	0.584	F	0.846
T	T	0.462	F	0.677	F	0.769	F	0.739	F	0.584	F	0.8
F	T	0.375	F	0.93	F	0.996	F	0.837	F	0.557	T	0.414
T	T	0.375	F	0.693	T	0.002	T	0.213	F	0.545	T	0.414
F	T	0.375	F	0.549	T	0.318	F	0.748	T	0.45	F	0.615
T	F	0.896	F	0.925	T	0.272	F	0.795	T	0.469	T	0.414
F	F	0.896	F	0.925	T	0.272	F	0.795	T	0.469	T	0.414
T	F	0.896	F	0.925	T	0.272	F	0.795	T	0.469	T	0.414
T	T	0.375	T	0.032	T	0.017	T	0.049	F	0.557	F	0.615
F	T	0.375	T	0.342	T	0.01	F	0.634	F	0.557	F	0.615
F	F	0.98	T	0.002	T	0.11	T	0.02	F	0.557	F	1
T	T	0.375	T	0.427	F	0.781	F	0.742	F	0.563	F	0.615
T	T	0.375	T	0	T	0.096	T	0.019	F	0.557	T	0.414
F	T	0.375	T	0.267	F	0.714	T	0.266	F	0.557	F	0.615
F	T	0.375	F	0.605	F	0.978	F	0.739	F	0.557	F	0.615
F	T	0.375	F	0.849	F	0.999	F	0.816	F	0.573	T	0.414
F	F	0.507	F	0.882	F	0.937	F	0.722	F	0.714	T	0.382
F	F	0.507	F	0.928	T	0.144	T	0.388	T	0.227	T	0.382
F	F	0.985	F	0.946	F	0.999	F	0.664	F	0.832	F	1
F	F	0.507	F	0.996	F	0.967	F	0.664	F	0.551	F	1
F	F	0.507	T	0.019	T	0.157	T	0.145	F	0.551	T	0.111
F	T	0.023	T	0.003	T	0.119	T	0.114	F	0.551	T	0
T	F	0.507	T	0.019	T	0.422	T	0.137	F	0.551	T	0.111
F	T	0.023	T	0.005	T	0.049	T	0.116	F	0.551	T	0
F	T	0.023	T	0.005	T	0.049	T	0.116	F	0.551	T	0
F	F	0.507	T	0.144	T	0.169	T	0.342	F	0.551	T	0.111
F	F	0.507	T	0.302	T	0.213	F	0.538	F	0.551	F	0.708

F	F	0.507	F	0.655	F	0.786	F	0.618	F	0.551	F	0.708
F	T	0.023	T	0.107	T	0.056	T	0.233	F	0.551	T	0
F	F	0.507	T	0.161	T	0.159	T	0.357	F	0.544	T	0.111
F	T	0.476	F	0.675	F	0.745	F	0.735	F	0.641	F	0.603
F	F	0.984	F	0.831	F	0.97	F	0.733	F	0.595	F	1
F	F	0.984	F	0.941	F	0.931	F	0.778	F	0.61	F	1
F	F	0.984	F	0.938	F	0.942	F	0.736	F	0.598	F	1
F	F	0.984	F	0.997	F	0.991	T	0.091	F	0.797	F	1
T	T	0.476	T	0.243	T	0.104	F	0.646	F	0.595	F	0.603
F	T	0.476	F	0.664	F	0.766	F	0.728	F	0.595	F	0.603
F	T	0.476	T	0	T	0.066	T	0.052	F	0.595	T	0
T	T	0.476	T	0.008	T	0.003	T	0.066	F	0.595	T	0
T	T	0.476	T	0.149	T	0.007	T	0.186	F	0.595	F	0.603
T	T	0.476	T	0.215	T	0.007	T	0.308	F	0.595	F	0.603
T	T	0.476	T	0.359	T	0.144	F	0.69	F	0.631	T	0.071
T	T	0.476	F	0.816	T	0.166	T	0.474	F	0.585	F	0.603
T	T	0.476	F	0.884	T	0.169	F	0.507	T	0.485	F	0.603
T	F	0.501	F	0.635	F	0.833	F	0.756	F	0.611	F	0.607
T	F	0.501	F	0.952	T	0.449	F	0.75	T	0.446	F	0.607
T	F	0.501	F	0.968	T	0.489	F	0.768	F	0.816	F	0.607
F	F	0.501	F	0.89	F	0.62	F	0.731	F	0.611	F	0.607
F	F	0.985	F	0.968	F	0.988	F	0.776	F	0.617	F	1
T	F	0.501	T	0.196	T	0.003	F	0.697	F	0.611	T	0
T	F	0.501	T	0.174	T	0.001	F	0.534	F	0.611	F	0.737
T	F	0.501	T	0.277	T	0.003	F	0.699	F	0.611	T	0
T	F	0.501	T	0.344	T	0.28	F	0.684	F	0.611	T	0.111
T	F	0.501	F	0.76	T	0.239	F	0.569	T	0.248	F	0.607
F	F	0.501	F	0.703	F	0.875	F	0.776	F	0.611	T	0.111
T	F	0.501	F	0.567	T	0.003	T	0.175	F	0.611	T	0
T	F	0.501	F	0.825	T	0.239	F	0.691	F	0.612	F	0.607
T	F	0.501	F	0.858	T	0.065	F	0.632	F	0.602	F	0.607
F	T	0.425	F	0.784	F	0.784	F	0.665	F	0.56	F	0.56
F	T	0.425	T	0.367	T	0.367	F	0.748	F	0.577	T	0.1
F	T	0.425	F	0.81	F	0.81	F	0.534	F	0.555	F	0.556
F	F	0.985	F	0.905	F	0.905	F	0.748	F	0.568	F	1
T	T	0.425	T	0.166	T	0.166	F	0.655	F	0.553	F	0.556
F	T	0.425	T	0.238	T	0.238	F	0.744	F	0.555	F	0.556
T	T	0.425	T	0.409	T	0.409	T	0.445	F	0.555	F	0.556
F	T	0.425	T	0.377	T	0.377	F	0.747	F	0.554	F	0.556
T	T	0.425	F	0.708	F	0.708	F	0.746	F	0.555	F	0.556
F	T	0.425	F	0.989	F	0.989	T	0.469	T	0.446	T	0.105

F	T	0.425	T	0.29	T	0.29	F	0.714	F	0.555	F	0.556
F	T	0.425	F	0.979	F	0.979	F	0.748	F	0.555	F	0.556
F	F	0.985	F	0.969	F	0.969	F	0.773	F	0.555	F	1
F	T	0.425	F	0.787	F	0.787	F	0.767	F	0.559	F	0.56
F	T	0.393	F	0.565	F	0.776	F	0.799	F	0.56	T	0.2
F	T	0.393	F	0.511	T	0.062	F	0.523	F	0.629	T	0.059
F	T	0.393	T	0.451	F	0.504	F	0.802	F	0.559	F	1
T	F	0.843	F	0.907	T	0.198	F	0.741	T	0.402	T	0.3
F	F	0.978	F	0.992	F	0.973	F	0.521	F	0.727	F	1
F	F	0.997	F	0.94	F	0.994	F	0.719	F	0.841	F	1
F	F	0.997	F	0.94	F	0.994	F	0.719	F	0.841	F	1
T	T	0.393	T	0.178	T	0.364	F	0.722	F	0.559	F	1
T	T	0.393	T	0.317	T	0.004	T	0.347	T	0.251	T	0.059
F	T	0.393	T	0.392	T	0.005	T	0.269	F	0.557	T	0.059
F	F	0.978	F	0.899	F	0.982	F	0.811	F	0.559	F	1
T	T	0.393	T	0.465	F	0.94	F	0.746	F	0.659	F	1
F	F	0.978	F	0.876	F	0.999	F	0.775	F	0.577	F	1
T	T	0.393	T	0.144	T	0.015	T	0.145	F	0.559	F	0.818
T	T	0.453	T	0.459	T	0.024	F	0.516	F	0.544	T	0.125
T	T	0.453	F	0.71	T	0.328	T	0.418	T	0.34	T	0.125
F	T	0.453	F	0.85	F	0.974	F	0.773	F	0.561	F	0.769
T	T	0.453	F	0.722	F	0.944	F	0.771	F	0.546	F	0.71
F	T	0.453	F	0.733	F	0.91	F	0.666	F	0.534	F	0.71
F	T	0.453	F	0.878	F	0.736	T	0.314	F	0.548	F	0.769
F	F	0.978	F	0.935	F	0.998	F	0.666	F	0.554	F	1
F	T	0.453	F	0.837	F	1	F	0.77	F	0.548	F	0.71
F	F	0.978	F	0.932	F	0.997	F	0.624	F	0.554	F	1
F	F	0.978	F	0.898	F	0.839	F	0.719	F	0.549	F	1
F	F	0.978	F	0.88	F	0.994	F	0.772	F	0.781	F	1
F	F	0.978	F	0.914	F	0.941	F	0.649	F	0.671	F	1
F	F	0.978	F	0.914	F	0.941	F	0.649	F	0.671	F	1
F	F	0.978	F	0.939	F	0.978	F	0.772	F	0.548	F	1
F	F	0.978	F	0.906	F	0.996	F	0.667	F	0.781	F	1
F	F	0.978	T	0.009	F	0.999	T	0.182	F	0.548	F	1
F	F	0.978	F	0.591	F	1	F	0.667	F	0.565	F	1
F	F	0.978	F	0.855	F	0.611	F	0.773	F	0.548	F	1
F	F	0.978	F	0.935	F	0.998	F	0.666	F	0.554	F	1

	Linear Ensemble			
	Majority Voting	Average Probability	Best	Weight

Actual				Accuracy		Accuracy	Accuracy
T	T	0.370333333	T	SVM	0	F	T
T	T	0.3795	T		0	F	T
T	T	0.342833333	T		0.125	F	T
T	T	0.382333333	T		0	F	T
T	T	0.311333333	T		0	F	T
T	T	0.453833333	T		0.125	F	T
T	T	0.418833333	T		0.727	F	T
T	T	0.348666667	T		0	F	T
T	F	0.796666667	F		0.857	F	F
T	F	0.796666667	F		0.857	F	F
T	F	0.868	F		0.857	F	F
F	F	0.737833333	F		1	F	F
T	T	0.501833333	T		0.3	F	T
F	F	0.868833333	F		0.857	F	F
F	T	0.413666667	T	SVM	0.414	T	T
T	T	0.268333333	T		0.056	F	T
T	T	0.28	T		0.056	F	T
F	F	0.7255	F		1	F	F
F	F	0.941833333	F		1	F	F
F	F	0.6805	F		0.414	F	F
T	T	0.576333333	F		0.414	F	T
F	F	0.9475	F		1	F	F
F	F	0.9475	F		1	F	F
F	T	0.5285	T		0.414	F	T
T	T	0.251666667	T		0.056	T	T
T	T	0.503333333	T		0.414	F	T
T	T	0.499	T		0.414	F	T
T	F	0.7815	F		0.414	F	F
T	F	0.588666667	F	SVM	0	F	T
F	F	0.937333333	F		1	F	F
F	F	0.938	F		1	F	F
T	F	0.853333333	F		0.846	F	F
T	F	0.872	F		1	F	F
F	F	0.8565	F		1	F	F
F	F	0.735666667	F		0.846	F	F
F	F	0.9835	F		1	F	F
F	F	0.823666667	F		0.8	F	F
F	F	0.984333333	F		1	F	F
F	F	0.9615	F		1	F	F
T	T	0.413333333	T		0.167	F	T

F	F	0.741833333	F	SVM	0.846	F	F
T	F	0.741166667	F		0.8	F	F
F	F	0.758666667	F		0.414	F	F
T	T	0.4495	T		0.414	F	T
F	T	0.434166667	F		0.615	T	F
T	T	0.550333333	F		0.414	T	F
F	T	0.550333333	F		0.414	T	F
T	T	0.550333333	F		0.414	T	F
T	T	0.348	T		0.615	F	T
F	T	0.496	T		0.615	F	T
F	T	0.518666667	T		1	F	T
T	F	0.656666667	F		0.615	F	F
T	T	0.317333333	T		0.414	F	T
F	T	0.5395	T		0.615	F	F
F	F	0.718666667	F		0.615	F	F
F	F	0.742166667	F	SVM	0.414	F	F
F	F	0.738333333	F		0.382	F	F
F	T	0.3915	T		0.382	T	T
F	F	0.932333333	F		1	F	F
F	F	0.855666667	F		1	F	F
F	T	0.323166667	T		0.111	F	T
F	T	0.209833333	T		0	F	T
T	T	0.366	T		0.111	F	T
F	T	0.198833333	T		0	F	T
F	T	0.198833333	T		0	F	T
F	T	0.378833333	T		0.111	F	T
F	F	0.544666667	T		0.708	F	T
F	F	0.712333333	F		0.708	F	F
F	T	0.2365	T		0	F	T
F	T	0.3825	T		0.111	F	T
F	F	0.705666667	F	SVM	0.603	F	F
F	F	0.919666667	F		1	F	F
F	F	0.939	F		1	F	F
F	F	0.933333333	F		1	F	F
F	F	0.843833333	F		1	F	F
T	T	0.512	T		0.603	F	T
F	F	0.706166667	F		0.603	F	F
F	T	0.265666667	T		0	F	T
T	T	0.258833333	T		0	F	T
T	T	0.4035	T		0.603	F	T
T	T	0.434833333	T		0.603	F	T

T	T	0.456666667	T		0.071	F	T
T	T	0.589166667	F		0.603	F	T
T	T	0.439833333	F		0.603	T	T
T	F	0.722	F	SVM	0.607	F	F
T	F	0.709833333	F		0.607	T	F
T	F	0.722166667	F		0.607	F	F
F	F	0.724833333	F		0.607	F	F
F	F	0.952833333	F		1	F	F
T	T	0.3995	T		0	F	T
T	F	0.491166667	T		0.737	F	T
T	T	0.413333333	T		0	F	T
T	T	0.486666667	T		0.111	F	T
T	F	0.446	T		0.607	T	T
F	F	0.661	F		0.111	F	F
T	T	0.374333333	T		0	F	T
T	F	0.643833333	F		0.607	F	F
T	F	0.6105	F		0.607	F	T
F	F	0.703	F	SVM	0.56	F	F
F	T	0.501166667	T		0.1	F	T
F	F	0.689166667	F		0.556	F	F
F	F	0.923833333	F		1	F	F
T	T	0.494666667	T		0.556	F	T
F	T	0.5335	T		0.556	F	T
T	T	0.540666667	T		0.556	F	T
F	T	0.580333333	F		0.556	F	F
T	F	0.6905	F		0.556	F	F
F	T	0.496166667	F		0.105	T	F
F	T	0.545833333	T		0.556	F	T
F	F	0.781166667	F		0.556	F	F
F	F	0.949333333	F		1	F	F
F	F	0.721	F		0.56	F	F
F	F	0.622166667	F	DT	0.2	T	F
F	T	0.424666667	T		0.059	T	T
F	F	0.691666667	F		1	F	F
T	T	0.498166667	F		0.3	T	F
F	F	0.910666667	F		1	F	F
F	F	0.941666667	F		1	F	F
F	F	0.941666667	F		1	F	F
T	T	0.6095	F		1	F	F
T	T	0.186666667	T		0.059	T	T
F	T	0.353	T		0.059	T	T

F	F	0.945	F	SVM	1	F	F
T	F	0.757333333	F		1	F	F
F	F	0.938	F		1	F	F
T	T	0.419166667	T		0.818	F	T
T	T	0.4295	T		0.544	T	T
T	T	0.339	T		0.34	T	T
F	F	0.803166667	F		0.561	F	F
T	F	0.766666667	F		0.546	F	F
F	F	0.745333333	F		0.534	F	F
F	F	0.691666667	F		0.548	F	F
F	F	0.9295	F		0.554	F	F
F	F	0.795	F		0.548	F	F
F	F	0.921833333	F		0.554	F	F
F	F	0.905666667	F		0.549	F	F
F	F	0.937333333	F		0.781	F	F
F	F	0.913666667	F		0.671	F	F
F	F	0.913666667	F		0.671	F	F
F	F	0.9445	F		0.548	F	F
F	F	0.9245	F		0.781	F	F
F	F	0.694666667	F		0.548	F	F
F	F	0.872666667	F		0.565	F	F
F	F	0.8695	F		0.548	F	F
F	F	0.9295	F		0.554	F	F

BBN	NB	MLP	RBF	SVM	DT	
Predicted	Predicted	Predicted	Predicted	Predicted	Predicted	Majority Voting
F	T	T	T	F	T	T
F	T	T	T	F	T	T
T	T	T	T	F	T	T
F	T	T	T	F	T	T
T	T	T	T	F	T	T
T	T	T	F	F	T	T
T	T	T	T	F	F	T
T	T	T	T	F	T	T
F	F	T	F	F	F	F
F	F	T	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	T	T	F	T	T
T	F	F	F	F	F	F
T	F	T	F	T	T	T

T	T	T	T	F	T	T
T	T	T	T	F	T	T
T	F	F	F	F	F	F
F	F	F	F	F	F	F
T	F	F	F	F	T	F
T	F	T	F	F	T	T
F	F	F	F	F	F	F
F	F	F	F	F	F	F
T	F	T	F	F	T	T
T	T	T	F	T	T	T
T	T	T	F	F	T	T
T	T	T	F	F	T	T
T	F	F	F	F	T	F
T	F	F	F	F	T	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	T	F	F	F	F
F	F	F	F	F	F	F
T	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
T	T	T	F	F	T	T
T	F	T	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	T	F
T	F	T	T	F	T	T
T	F	T	F	T	F	T
F	F	T	F	T	T	T
F	F	T	F	T	T	T
T	T	T	T	F	F	T
T	T	T	F	F	F	T
F	T	T	T	F	F	T
T	T	F	F	F	F	F
T	T	T	T	F	T	T
T	T	F	T	F	F	T
T	F	F	F	F	F	F
T	F	F	F	F	T	F

F	F	F	F	F	T	F
F	F	T	T	T	T	T
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	T	T	T	F	T	T
T	T	T	T	F	T	T
F	T	T	T	F	T	T
T	T	T	T	F	T	T
T	T	T	T	F	T	T
F	T	T	T	F	T	T
F	T	T	F	F	F	F
F	F	F	F	F	F	F
T	T	T	T	F	T	T
F	T	T	T	F	T	T
T	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	T	F	F	F
T	T	T	F	F	F	T
T	F	F	F	F	F	F
T	T	T	T	F	T	T
T	T	T	T	F	T	T
T	T	T	T	F	F	T
T	T	T	T	F	F	T
T	T	T	F	F	T	T
T	F	T	T	F	F	T
T	F	T	F	T	F	T
F	F	F	F	F	F	F
F	F	T	F	T	F	F
F	F	T	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	T	T	F	F	T	T
F	T	T	F	F	F	F
F	T	T	F	F	T	T
F	T	T	F	F	T	T
F	F	T	F	T	F	F
F	F	F	F	F	T	F
F	F	T	T	F	T	T
F	F	T	F	F	F	F

F	F	T	F	F	F	F
T	F	F	F	F	F	F
T	T	T	F	F	T	T
T	F	F	F	F	F	F
F	F	F	F	F	F	F
T	T	T	F	F	F	T
T	T	T	F	F	F	T
T	T	T	T	F	F	T
T	T	T	F	F	F	T
T	F	F	F	F	F	F
T	F	F	T	T	T	T
T	T	T	F	F	F	T
T	F	F	F	F	F	F
F	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	T	F
T	F	T	F	F	T	T
T	T	F	F	F	F	F
F	F	T	F	T	T	T
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
T	T	T	F	F	F	T
T	T	T	T	T	T	T
T	T	T	T	F	T	T
F	F	F	F	F	F	F
T	T	F	F	F	F	F
F	F	F	F	F	F	F
T	T	T	T	F	F	T
T	T	T	F	F	T	T
T	F	T	T	T	T	T
T	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	F	F	F	F	F
T	F	F	T	F	F	F
F	F	F	F	F	F	F
T	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F

F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	T	F	T	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F
F	F	F	F	F	F	F

BBN	NB	MLP	RBF	SVM	DT		
Probability	Probability	Probability	Probability	Probability	Probability	Average prob	
0.914	0.003	0	0.305	0.619	0	0.306833333	T
0.914	0.004	0.055	0.304	0.619	0	0.316	T
0.42	0.064	0.078	0.37	0.619	0.125	0.279333333	T
0.914	0.019	0.053	0.308	0.619	0	0.318833333	T
0.42	0.11	0	0.338	0.619	0	0.247833333	T
0.42	0.448	0.202	0.528	0.747	0.125	0.411666667	T
0.42	0.007	0.055	0.304	0.619	0.727	0.355333333	T
0.42	0.274	0.012	0.386	0.618	0	0.285	T
0.914	0.984	0.089	0.936	0.619	0.857	0.733166667	F
0.914	0.984	0.089	0.936	0.619	0.857	0.733166667	F
0.42	1	0.992	0.939	0.612	0.857	0.803333333	F
0.42	0.628	0.746	0.633	0.619	1	0.674333333	F
0.42	0.774	0.172	0.345	0.615	0.3	0.437666667	T
0.42	1	1	0.936	0.62	0.857	0.8055	F
0.471	0.876	0.146	0.575	0.2	0.414	0.447	T
0.471	0.001	0.001	0.081	0.588	0.056	0.199666667	T
0.471	0.048	0.008	0.097	0.588	0.056	0.211333333	T
0.471	0.511	0.647	0.724	0.588	1	0.656833333	F
0.984	0.876	0.98	0.811	0.588	1	0.873166667	F
0.471	0.79	0.79	0.618	0.642	0.414	0.620833333	F
0.471	0.846	0.186	0.541	0.598	0.414	0.509333333	F
0.984	0.962	1	0.739	0.787	1	0.912	F
0.984	0.962	1	0.739	0.787	1	0.912	F
0.471	0.505	0.055	0.726	0.588	0.414	0.459833333	T
0.471	0.372	0.01	0.601	0.357	0.056	0.311166667	T
0.471	0.398	0.015	0.722	0.588	0.414	0.434666667	T
0.471	0.408	0.02	0.681	0.588	0.414	0.430333333	T
0.471	0.972	0.995	0.837	0.614	0.414	0.717166667	F
0.462	0.507	0.827	0.736	0.584	0	0.519333333	F
0.983	0.821	0.847	0.973	0.584	1	0.868	F

0.983	0.93	0.788	0.927	0.697	1	0.8875	F
0.462	0.863	1	0.949	0.583	0.846	0.783833333	F
0.462	0.91	0.999	0.861	0.584	1	0.802666667	F
0.462	0.827	0.983	0.867	0.533	1	0.778666667	F
0.462	0.829	0.302	0.975	0.548	0.846	0.660333333	F
0.983	0.968	1	0.95	0.692	1	0.932166667	F
0.462	0.887	1	0.793	0.587	0.8	0.754833333	F
0.983	0.954	1	0.969	0.584	1	0.915	F
0.983	0.9	0.997	0.889	0.796	1	0.9275	F
0.462	0.236	0.018	0.597	0.584	0.167	0.344	T
0.462	0.753	0.415	0.975	0.584	0.846	0.6725	F
0.462	0.677	0.769	0.739	0.584	0.8	0.671833333	F
0.375	0.93	0.996	0.837	0.557	0.414	0.684833333	F
0.375	0.693	0.002	0.213	0.545	0.414	0.373666667	T
0.375	0.549	0.318	0.748	0.45	0.615	0.509166667	F
0.896	0.925	0.272	0.795	0.469	0.414	0.6285	F
0.896	0.925	0.272	0.795	0.469	0.414	0.6285	F
0.896	0.925	0.272	0.795	0.469	0.414	0.6285	F
0.375	0.032	0.017	0.049	0.557	0.615	0.274166667	T
0.375	0.342	0.01	0.634	0.557	0.615	0.422166667	T
0.98	0.002	0.11	0.02	0.557	1	0.444833333	T
0.375	0.427	0.781	0.742	0.563	0.615	0.583833333	F
0.375	0	0.096	0.019	0.557	0.414	0.2435	T
0.375	0.267	0.714	0.266	0.557	0.615	0.465666667	T
0.375	0.605	0.978	0.739	0.557	0.615	0.644833333	F
0.375	0.849	0.999	0.816	0.573	0.414	0.671	F
0.507	0.882	0.937	0.722	0.714	0.382	0.690666667	F
0.507	0.928	0.144	0.388	0.227	0.382	0.429333333	T
0.985	0.946	0.999	0.664	0.832	1	0.904333333	F
0.507	0.996	0.967	0.664	0.551	1	0.780833333	F
0.507	0.019	0.157	0.145	0.551	0.111	0.248333333	T
0.023	0.003	0.119	0.114	0.551	0	0.135	T
0.507	0.019	0.422	0.137	0.551	0.111	0.291166667	T
0.023	0.005	0.049	0.116	0.551	0	0.124	T
0.023	0.005	0.049	0.116	0.551	0	0.124	T
0.507	0.144	0.169	0.342	0.551	0.111	0.304	T
0.507	0.302	0.213	0.538	0.551	0.708	0.469833333	T
0.507	0.655	0.786	0.618	0.551	0.708	0.6375	F
0.023	0.107	0.056	0.233	0.551	0	0.161666667	T
0.507	0.161	0.159	0.357	0.544	0.111	0.3065	T
0.476	0.675	0.745	0.735	0.641	0.603	0.645833333	F

0.984	0.831	0.97	0.733	0.595	1	0.852166667	F
0.984	0.941	0.931	0.778	0.61	1	0.874	F
0.984	0.938	0.942	0.736	0.598	1	0.866333333	F
0.984	0.997	0.991	0.091	0.797	1	0.81	F
0.476	0.243	0.104	0.646	0.595	0.603	0.4445	T
0.476	0.664	0.766	0.728	0.595	0.603	0.638666667	F
0.476	0	0.066	0.052	0.595	0	0.198166667	T
0.476	0.008	0.003	0.066	0.595	0	0.191333333	T
0.476	0.149	0.007	0.186	0.595	0.603	0.336	T
0.476	0.215	0.007	0.308	0.595	0.603	0.367333333	T
0.476	0.359	0.144	0.69	0.631	0.071	0.395166667	T
0.476	0.816	0.166	0.474	0.585	0.603	0.52	F
0.476	0.884	0.169	0.507	0.485	0.603	0.520666667	F
0.501	0.635	0.833	0.756	0.611	0.607	0.657166667	F
0.501	0.952	0.449	0.75	0.446	0.607	0.6175	F
0.501	0.968	0.489	0.768	0.816	0.607	0.6915	F
0.501	0.89	0.62	0.731	0.611	0.607	0.66	F
0.985	0.968	0.988	0.776	0.617	1	0.889	F
0.501	0.196	0.003	0.697	0.611	0	0.334666667	T
0.501	0.174	0.001	0.534	0.611	0.737	0.426333333	T
0.501	0.277	0.003	0.699	0.611	0	0.3485	T
0.501	0.344	0.28	0.684	0.611	0.111	0.421833333	T
0.501	0.76	0.239	0.569	0.248	0.607	0.487333333	T
0.501	0.703	0.875	0.776	0.611	0.111	0.596166667	F
0.501	0.567	0.003	0.175	0.611	0	0.3095	T
0.501	0.825	0.239	0.691	0.612	0.607	0.579166667	F
0.501	0.858	0.065	0.632	0.602	0.607	0.544166667	F
0.425	0.784	0.784	0.665	0.56	0.56	0.629666667	F
0.425	0.367	0.367	0.748	0.577	0.1	0.430666667	T
0.425	0.81	0.81	0.534	0.555	0.556	0.615	F
0.985	0.905	0.905	0.748	0.568	1	0.851833333	F
0.425	0.166	0.166	0.655	0.553	0.556	0.420166667	T
0.425	0.238	0.238	0.744	0.555	0.556	0.459333333	T
0.425	0.409	0.409	0.445	0.555	0.556	0.4665	T
0.425	0.377	0.377	0.747	0.554	0.556	0.506	F
0.425	0.708	0.708	0.746	0.555	0.556	0.616333333	F
0.425	0.989	0.989	0.469	0.446	0.105	0.5705	F
0.425	0.29	0.29	0.714	0.555	0.556	0.471666667	T
0.425	0.979	0.979	0.748	0.555	0.556	0.707	F
0.985	0.969	0.969	0.773	0.555	1	0.875166667	F
0.425	0.787	0.787	0.767	0.559	0.56	0.6475	F

0.393	0.565	0.776	0.799	0.56	0.2	0.548833333	F
0.393	0.511	0.062	0.523	0.629	0.059	0.362833333	T
0.393	0.451	0.504	0.802	0.559	1	0.618166667	F
0.843	0.907	0.198	0.741	0.402	0.3	0.565166667	F
0.978	0.992	0.973	0.521	0.727	1	0.865166667	F
0.997	0.94	0.994	0.719	0.841	1	0.915166667	F
0.997	0.94	0.994	0.719	0.841	1	0.915166667	F
0.393	0.178	0.364	0.722	0.559	1	0.536	F
0.393	0.317	0.004	0.347	0.251	0.059	0.2285	T
0.393	0.392	0.005	0.269	0.557	0.059	0.279166667	T
0.978	0.899	0.982	0.811	0.559	1	0.8715	F
0.393	0.465	0.94	0.746	0.659	1	0.7005	F
0.978	0.876	0.999	0.775	0.577	1	0.8675	F
0.393	0.144	0.015	0.145	0.559	0.818	0.345666667	T
0.453	0.459	0.024	0.516	0.544	0.125	0.3535	T
0.453	0.71	0.328	0.418	0.34	0.125	0.395666667	T
0.453	0.85	0.974	0.773	0.561	0.769	0.73	F
0.453	0.722	0.944	0.771	0.546	0.71	0.691	F
0.453	0.733	0.91	0.666	0.534	0.71	0.667666667	F
0.453	0.878	0.736	0.314	0.548	0.769	0.616333333	F
0.978	0.935	0.998	0.666	0.554	1	0.855166667	F
0.453	0.837	1	0.77	0.548	0.71	0.719666667	F
0.978	0.932	0.997	0.624	0.554	1	0.8475	F
0.978	0.898	0.839	0.719	0.549	1	0.8305	F
0.978	0.88	0.994	0.772	0.781	1	0.900833333	F
0.978	0.914	0.941	0.649	0.671	1	0.858833333	F
0.978	0.914	0.941	0.649	0.671	1	0.858833333	F
0.978	0.939	0.978	0.772	0.548	1	0.869166667	F
0.978	0.906	0.996	0.667	0.781	1	0.888	F
0.978	0.009	0.999	0.182	0.548	1	0.619333333	F
0.978	0.591	1	0.667	0.565	1	0.800166667	F
0.978	0.855	0.611	0.773	0.548	1	0.794166667	F
0.978	0.935	0.998	0.666	0.554	1	0.855166667	F

	Train set#									
Accuracy	1	2	3	4	5	6	7	8	9	10
BBN	66.412 21	64.122 14	64.122 14	71.755 73	70.992 37	63.358 78	62.595 42	67.938 93	69.465 65	62.698 41
NB	65.648 85	67.938 93	70.992 37	73.282 44	70.992 37	67.938 93	72.519 08	71.755 73	71.755 73	69.841 27
MLP	90	89.312	88.549	93.893	91.603	89.312	90.076	91.603	90.076	87.301

		98	62	13	05	98	34	05	34	59
RBF	71.755 73	71.755 73	80.152 67	77.862 6	76.335 88	77.862 6	72.519 08	75.572 52	71.755 73	71.428 57
SVM	97.709 92	98.473 28	97.709 92	97.709 92	98.473 28	97.709 92	90.839 69	96.946 56	64.122 14	96.825 4
DT	85.496 18	82.442 75	89.312 98	77.099 24	83.206 11	77.099 24	78.625 95	74.045 8	90.839 69	83.333 33

	Train set#									
Ranking	1	2	3	4	5	6	7	8	9	10
BBN	2	1	1	1	1	1	1	1	2	1
NB	1	2	2	2	2	2	2	2	3	2
MLP	5	5	4	5	5	5	5	5	5	5
RBF	3	3	3	4	3	4	3	4	4	3
SVM	6	6	6	6	6	6	6	6	1	6
DT	4	4	5	3	4	3	4	3	6	4

BBN	Weight	NB	Weight	MLP	Weight	RBF	Weight	SVM	Weight	DT	Weight	Total	Prediction
0.914	2	0.003	1	0	5	0.305	3	0.619	6	0	4	0.307619	T
0.914	2	0.004	1	0.055	5	0.304	3	0.619	6	0	4	0.320619	T
0.42	2	0.064	1	0.078	5	0.37	3	0.619	6	0.125	4	0.315143	T
0.914	2	0.019	1	0.053	5	0.308	3	0.619	6	0	4	0.321429	T
0.42	2	0.11	1	0	5	0.338	3	0.619	6	0	4	0.270381	T
0.42	2	0.448	1	0.202	5	0.528	3	0.747	6	0.125	4	0.422095	T
0.42	2	0.007	1	0.055	5	0.304	3	0.619	6	0.727	4	0.41219	T
0.42	2	0.274	1	0.012	5	0.386	3	0.618	6	0	4	0.287619	T
0.914	2	0.984	1	0.089	5	0.936	3	0.619	6	0.857	4	0.628905	F
0.914	2	0.984	1	0.089	5	0.936	3	0.619	6	0.857	4	0.628905	F
0.42	2	1	1	0.992	5	0.939	3	0.612	6	0.857	4	0.796048	F
0.42	2	0.628	1	0.746	5	0.633	3	0.619	6	1	4	0.705286	F
0.42	2	0.774	1	0.172	5	0.345	3	0.615	6	0.3	4	0.399952	T
0.42	2	1	1	1	5	0.93	3	0.62	6	0.85	4	0.79	F

						6				7		981	
0.47 1	1	0.87 6	2	0.14 6	5	0.57 5	3	0.2	6	0.41 4	4	0.35 8762	T
0.47 1	1	0.00 1	2	0.00 1	5	0.08 1	3	0.58 8	6	0.05 6	4	0.21 3	T
0.47 1	1	0.04 8	2	0.00 8	5	0.09 7	3	0.58 8	6	0.05 6	4	0.22 1429	T
0.47 1	1	0.51 1	2	0.64 7	5	0.72 4	3	0.58 8	6	1	4	0.68 7048	F
0.98 4	1	0.87 6	2	0.98	5	0.81 1	3	0.58 8	6	1	4	0.83 7952	F
0.47 1	1	0.79	2	0.79	5	0.61 8	3	0.64 2	6	0.41 4	4	0.63 6333	F
0.47 1	1	0.84 6	2	0.18 6	5	0.54 1	3	0.59 8	6	0.41 4	4	0.47 4286	T
0.98 4	1	0.96 2	2	1	5	0.73 9	3	0.78 7	6	1	4	0.89 7476	F
0.98 4	1	0.96 2	2	1	5	0.73 9	3	0.78 7	6	1	4	0.89 7476	F
0.47 1	1	0.50 5	2	0.05 5	5	0.72 6	3	0.58 8	6	0.41 4	4	0.43 419	T
0.47 1	1	0.37 2	2	0.01	5	0.60 1	3	0.35 7	6	0.05 6	4	0.25 8762	T
0.47 1	1	0.39 8	2	0.01 5	5	0.72 2	3	0.58 8	6	0.41 4	4	0.41 3905	T
0.47 1	1	0.40 8	2	0.02	5	0.68 1	3	0.58 8	6	0.41 4	4	0.41 019	T
0.47 1	1	0.97 2	2	0.99 5	5	0.83 7	3	0.61 4	6	0.41 4	4	0.72 5762	F
0.46 2	1	0.50 7	2	0.82 7	4	0.73 6	3	0.58 4	6	0	5	0.49 981	T
0.98 3	1	0.82 1	2	0.84 7	4	0.97 3	3	0.58 4	6	1	5	0.83 0286	F
0.98 3	1	0.93	2	0.78 8	4	0.92 7	3	0.69 7	6	1	5	0.85 5143	F
0.46 2	1	0.86 3	2	1	4	0.94 9	3	0.58 3	6	0.84 6	5	0.79 8238	F
0.46 2	1	0.91	2	0.99 9	4	0.86 1	3	0.58 4	6	1	5	0.82 6905	F
0.46 2	1	0.82 7	2	0.98 3	4	0.86 7	3	0.53 3	6	1	5	0.80 2238	F
0.46 2	1	0.82 9	2	0.30 2	4	0.97 5	3	0.54 8	6	0.84 6	5	0.65 5762	F
0.98 3	1	0.96 8	2	1	4	0.95	3	0.69 2	6	1	5	0.90 1	F
0.46 2	1	0.88 7	2	1	4	0.79 3	3	0.58 7	6	0.8	5	0.76 8429	F
0.98 3	1	0.95 4	2	1	4	0.96 9	3	0.58 4	6	1	5	0.87 1524	F
0.98 3	1	0.9	2	0.99 7	4	0.88 9	3	0.79 6	6	1	5	0.91 4952	F

0.46 2	1	0.23 6	2	0.01 8	4	0.59 7	3	0.58 4	6	0.16 7	5	0.33 981	T
0.46 2	1	0.75 3	2	0.41 5	4	0.97 5	3	0.58 4	6	0.84 6	5	0.68 0333	F
0.46 2	1	0.67 7	2	0.76 9	4	0.73 9	3	0.58 4	6	0.8	5	0.69 5857	F
0.37 5	1	0.93	2	0.99 6	5	0.83 7	4	0.55 7	6	0.41 4	3	0.72 1286	F
0.37 5	1	0.69 3	2	0.00 2	5	0.21 3	4	0.54 5	6	0.41 4	3	0.33 9762	T
0.37 5	1	0.54 9	2	0.31 8	5	0.74 8	4	0.45	6	0.61 5	3	0.50 4762	F
0.89 6	1	0.92 5	2	0.27 2	5	0.79 5	4	0.46 9	6	0.41 4	3	0.54 0095	F
0.89 6	1	0.92 5	2	0.27 2	5	0.79 5	4	0.46 9	6	0.41 4	3	0.54 0095	F
0.89 6	1	0.92 5	2	0.27 2	5	0.79 5	4	0.46 9	6	0.41 4	3	0.54 0095	F
0.37 5	1	0.03 2	2	0.01 7	5	0.04 9	4	0.55 7	6	0.61 5	3	0.28 1286	T
0.37 5	1	0.34 2	2	0.01	5	0.63 4	4	0.55 7	6	0.61 5	3	0.42 0571	T
0.98	1	0.00 2	2	0.11	5	0.02	4	0.55 7	6	1	3	0.37 8857	T
0.37 5	1	0.42 7	2	0.78 1	5	0.74 2	4	0.56 3	6	0.61 5	3	0.63 4524	F
0.37 5	1	0	2	0.09 6	5	0.01 9	4	0.55 7	6	0.41 4	3	0.26 2619	T
0.37 5	1	0.26 7	2	0.71 4	5	0.26 6	4	0.55 7	6	0.61 5	3	0.51 0952	F
0.37 5	1	0.60 5	2	0.97 8	5	0.73 9	4	0.55 7	6	0.61 5	3	0.69 6095	F
0.37 5	1	0.84 9	2	0.99 9	5	0.81 6	4	0.57 3	6	0.41 4	3	0.71 4857	F
0.50 7	1	0.88 2	2	0.93 7	5	0.72 2	3	0.71 4	6	0.38 2	4	0.71 1143	F
0.50 7	1	0.92 8	2	0.14 4	5	0.38 8	3	0.22 7	6	0.38 2	4	0.33 9857	T
0.98 5	1	0.94 6	2	0.99 9	5	0.66 4	3	0.83 2	6	1	4	0.89 7905	F
0.50 7	1	0.99 6	2	0.96 7	5	0.66 4	3	0.55 1	6	1	4	0.79 2	F
0.50 7	1	0.01 9	2	0.15 7	5	0.14 5	3	0.55 1	6	0.11 1	4	0.26 2619	T
0.02 3	1	0.00 3	2	0.11 9	5	0.11 4	3	0.55 1	6	0	4	0.20 3429	T
0.50 7	1	0.01 9	2	0.42 2	5	0.13 7	3	0.55 1	6	0.11 1	4	0.32 4571	T
0.02 3	1	0.00 5	2	0.04 9	5	0.11 6	3	0.55 1	6	0	4	0.18 7238	T
0.02	1	0.00	2	0.04	5	0.11	3	0.55	6	0	4	0.18	T

3		5		9		6		1				7238	
0.50 7	1	0.14 4	2	0.16 9	5	0.34 2	3	0.55 1	6	0.11 1	4	0.30 5524	T
0.50 7	1	0.30 2	2	0.21 3	5	0.53 8	3	0.55 1	6	0.70 8	4	0.47 2762	T
0.50 7	1	0.65 5	2	0.78 6	5	0.61 8	3	0.55 1	6	0.70 8	4	0.65 4238	F
0.02 3	1	0.10 7	2	0.05 6	5	0.23 3	3	0.55 1	6	0	4	0.21 5333	T
0.50 7	1	0.16 1	2	0.15 9	5	0.35 7	3	0.54 4	6	0.11 1	4	0.30 4905	T
0.47 6	1	0.67 5	2	0.74 5	5	0.73 5	4	0.64 1	6	0.60 3	3	0.67 3619	F
0.98 4	1	0.83 1	2	0.97	5	0.73 3	4	0.59 5	6	1	3	0.80 9429	F
0.98 4	1	0.94 1	2	0.93 1	5	0.77 8	4	0.61	6	1	3	0.82 3476	F
0.98 4	1	0.93 8	2	0.94 2	5	0.73 6	4	0.59 8	6	1	3	0.81 4381	F
0.98 4	1	0.99 7	2	0.99 1	5	0.09 1	4	0.79 7	6	1	3	0.76 5667	F
0.47 6	1	0.24 3	2	0.10 4	5	0.64 6	4	0.59 5	6	0.60 3	3	0.44 9762	T
0.47 6	1	0.66 4	2	0.76 6	5	0.72 8	4	0.59 5	6	0.60 3	3	0.66 3095	F
0.47 6	1	0	2	0.06 6	5	0.05 2	4	0.59 5	6	0	3	0.21 8286	T
0.47 6	1	0.00 8	2	0.00 3	5	0.06 6	4	0.59 5	6	0	3	0.20 6714	T
0.47 6	1	0.14 9	2	0.00 7	5	0.18 6	4	0.59 5	6	0.60 3	3	0.33 0095	T
0.47 6	1	0.21 5	2	0.00 7	5	0.30 8	4	0.59 5	6	0.60 3	3	0.35 9619	T
0.47 6	1	0.35 9	2	0.14 4	5	0.69	4	0.63 1	6	0.07 1	3	0.41 3	T
0.47 6	1	0.81 6	2	0.16 6	5	0.47 4	4	0.58 5	6	0.60 3	3	0.48 3476	T
0.47 6	1	0.88 4	2	0.16 9	5	0.50 7	4	0.48 5	6	0.60 3	3	0.46 8381	T
0.50 1	1	0.63 5	2	0.83 3	5	0.75 6	3	0.61 1	6	0.60 7	4	0.68 0857	F
0.50 1	1	0.95 2	2	0.44 9	5	0.75	3	0.44 6	6	0.60 7	4	0.57 1619	F
0.50 1	1	0.96 8	2	0.48 9	5	0.76 8	3	0.81 6	6	0.60 7	4	0.69 0952	F
0.50 1	1	0.89	2	0.62	5	0.73 1	3	0.61 1	6	0.60 7	4	0.65 0857	F
0.98 5	1	0.96 8	2	0.98 8	5	0.77 6	3	0.61 7	6	1	4	0.85 1952	F
0.50 1	1	0.19 6	2	0.00 3	5	0.69 7	3	0.61 1	6	0	4	0.31 7381	T

0.50 1	1	0.17 4	2	0.00 1	5	0.53 4	3	0.61 1	6	0.73 7	4	0.43 1905	T
0.50 1	1	0.27 7	2	0.00 3	5	0.69 9	3	0.61 1	6	0	4	0.32 5381	T
0.50 1	1	0.34 4	2	0.28	5	0.68 4	3	0.61 1	6	0.11 1	4	0.41 6714	T
0.50 1	1	0.76	2	0.23 9	5	0.56 9	3	0.24 8	6	0.60 7	4	0.42 0905	T
0.50 1	1	0.70 3	2	0.87 5	5	0.77 6	3	0.61 1	6	0.11 1	4	0.60 5714	F
0.50 1	1	0.56 7	2	0.00 3	5	0.17 5	3	0.61 1	6	0	4	0.27 8143	T
0.50 1	1	0.82 5	2	0.23 9	5	0.69 1	3	0.61 2	6	0.60 7	4	0.54 8524	F
0.50 1	1	0.85 8	2	0.06 5	5	0.63 2	3	0.60 2	6	0.60 7	4	0.49 8952	T
0.42 5	1	0.78 4	2	0.78 4	5	0.66 5	4	0.56	6	0.56	3	0.64 8238	F
0.42 5	1	0.36 7	2	0.36 7	5	0.74 8	4	0.57 7	6	0.1	3	0.46 419	T
0.42 5	1	0.81	2	0.81	5	0.53 4	4	0.55 5	6	0.55 6	3	0.62 9952	F
0.98 5	1	0.90 5	2	0.90 5	5	0.74 8	4	0.56 8	6	1	3	0.79 619	F
0.42 5	1	0.16 6	2	0.16 6	5	0.65 5	4	0.55 3	6	0.55 6	3	0.43 7762	T
0.42 5	1	0.23 8	2	0.23 8	5	0.74 4	4	0.55 5	6	0.55 6	3	0.47 9286	T
0.42 5	1	0.40 9	2	0.40 9	5	0.44 5	4	0.55 5	6	0.55 6	3	0.47 9333	T
0.42 5	1	0.37 7	2	0.37 7	5	0.74 7	4	0.55 4	6	0.55 6	3	0.52 5905	F
0.42 5	1	0.70 8	2	0.70 8	5	0.74 6	4	0.55 5	6	0.55 6	3	0.63 6333	F
0.42 5	1	0.98 9	2	0.98 9	5	0.46 9	4	0.44 6	6	0.10 5	3	0.58 1667	F
0.42 5	1	0.29	2	0.29	5	0.71 4	4	0.55 5	6	0.55 6	3	0.49 0905	T
0.42 5	1	0.97 9	2	0.97 9	5	0.74 8	4	0.55 5	6	0.55 6	3	0.72 7048	F
0.98 5	1	0.96 9	2	0.96 9	5	0.77 3	4	0.55 5	6	1	3	0.81 8571	F
0.42 5	1	0.78 7	2	0.78 7	5	0.76 7	4	0.55 9	6	0.56	3	0.66 8381	F
0.39 3	2	0.56 5	3	0.77 6	5	0.79 9	4	0.56	1	0.2	6	0.53 8905	F
0.39 3	2	0.51 1	3	0.06 2	5	0.52 3	4	0.62 9	1	0.05 9	6	0.27 1619	T
0.39 3	2	0.45 1	3	0.50 4	5	0.80 2	4	0.55 9	1	1	6	0.68 6952	F
0.84	2	0.90	3	0.19	5	0.74	4	0.40	1	0.3	6	0.50	F

3		7		8		1		2				3	
0.97 8	2	0.99 2	3	0.97 3	5	0.52 1	4	0.72 7	1	1	6	0.88 6095	F
0.99 7	2	0.94	3	0.99 4	5	0.71 9	4	0.84 1	1	1	6	0.92 8619	F
0.99 7	2	0.94	3	0.99 4	5	0.71 9	4	0.84 1	1	1	6	0.92 8619	F
0.39 3	2	0.17 8	3	0.36 4	5	0.72 2	4	0.55 9	1	1	6	0.59 9381	F
0.39 3	2	0.31 7	3	0.00 4	5	0.34 7	4	0.25 1	1	0.05 9	6	0.17 8571	T
0.39 3	2	0.39 2	3	0.00 5	5	0.26 9	4	0.55 7	1	0.05 9	6	0.18 9238	T
0.97 8	2	0.89 9	3	0.98 2	5	0.81 1	4	0.55 9	1	1	6	0.92 219	F
0.39 3	2	0.46 5	3	0.94	5	0.74 6	4	0.65 9	1	1	6	0.78 6857	F
0.97 8	2	0.87 6	3	0.99 9	5	0.77 5	4	0.57 7	1	1	6	0.91 6952	F
0.39 3	2	0.14 4	3	0.01 5	5	0.14 5	4	0.55 9	1	0.81 8	6	0.34 9524	T
0.45 3	1	0.45 9	2	0.02 4	5	0.51 6	3	0.54 4	6	0.12 5	4	0.32 3952	T
0.45 3	1	0.71	2	0.32 8	5	0.41 8	3	0.34	6	0.12 5	4	0.34 7952	T
0.45 3	1	0.85	2	0.97 4	5	0.77 3	3	0.56 1	6	0.76 9	4	0.75 1619	F
0.45 3	1	0.72 2	2	0.94 4	5	0.77 1	3	0.54 6	6	0.71	4	0.71 6476	F
0.45 3	1	0.73 3	2	0.91	5	0.66 6	3	0.53 4	6	0.71	4	0.69 1	F
0.45 3	1	0.87 8	2	0.73 6	5	0.31 4	3	0.54 8	6	0.76 9	4	0.62 8333	F
0.97 8	1	0.93 5	2	0.99 8	5	0.66 6	3	0.55 4	6	1	4	0.81 7143	F
0.45 3	1	0.83 7	2	1	5	0.77	3	0.54 8	6	0.71	4	0.74 119	F
0.97 8	1	0.93 2	2	0.99 7	5	0.62 4	3	0.55 4	6	1	4	0.81 0619	F
0.97 8	1	0.89 8	2	0.83 9	5	0.71 9	3	0.54 9	6	1	4	0.78 1905	F
0.97 8	1	0.88	2	0.99 4	5	0.77 2	3	0.78 1	6	1	4	0.89 0952	F
0.97 8	1	0.91 4	2	0.94 1	5	0.64 9	3	0.67 1	6	1	4	0.83 2571	F
0.97 8	1	0.91 4	2	0.94 1	5	0.64 9	3	0.67 1	6	1	4	0.83 2571	F
0.97 8	1	0.93 9	2	0.97 8	5	0.77 2	3	0.54 8	6	1	4	0.82 619	F
0.97 8	1	0.90 6	2	0.99 6	5	0.66 7	3	0.78 1	6	1	4	0.87 8905	F

0.97 8	1	0.00 9	2	0.99 9	5	0.18 2	3	0.54 8	6	1	4	0.65 8333	F
0.97 8	1	0.59 1	2	1	5	0.66 7	3	0.56 5	6	1	4	0.78 8143	F
0.97 8	1	0.85 5	2	0.61 1	5	0.77 3	3	0.54 8	6	1	4	0.73 0952	F
0.97 8	1	0.93 5	2	0.99 8	5	0.66 6	3	0.55 4	6	1	4	0.81 7143	F

KC1 (regression) results

Actual	MLP	RBF	SVMreg	M5P
8.13	3.036	32.609	20.209	9.981
13.1	-14.656	32.609	8.812	18.202
1.97	24.797	32.609	48.164	52.642
13.49	-3.47	32.609	14.62	14.913
11.9	32.002	32.609	24.625	22.719
7.43	35.319	32.609	28.765	33
10.24	6.999	23.713	22.424	22.195
7.87	17.064	23.731	24.987	26.036
16.53	-72.187	23.713	6.515	27.113
12.93	-72.187	23.713	6.515	27.113
18.66	14.302	23.713	-52.487	47.238
30.3	27.199	43.108	35.086	47.238
18.33	-10.493	23.511	20.67	27.432
30.82	10.502	23.511	21.475	8.498
12.58	35.34	42.261	30.673	41.587
38.67	33.814	42.035	26.7	23.205
31.25	29.531	25.536	22.808	30.558
26.22	35.365	42.261	27.248	28.719
140.13	-1.751	37.779	30.686	36.653
6.29	38.588	26.632	31.661	28.529
5.47	28.709	37.659	28.354	36.653
23.26	16.973	27.883	21.015	28.529
6.83	30.063	32.748	26.182	28.529
17.24	20.813	28.575	25.144	28.529
72.16	30.701	40.523	25.881	42.477
15.63	43.993	40.573	34.258	47.165
34.78	43.993	40.573	34.258	47.165
10.02	70.368	22.005	14.527	17.144
50.96	69.367	24.124	24.996	41.025
35.03	-20.434	22.005	-0.1	17.821
6.56	39.999	28.713	24.384	28.881
13.26	36.511	28.713	20.493	28.881
23.92	35.518	28.713	26.227	19.206
21.15	47.1	28.713	15.897	28.881
36.17	50.41	28.713	17.965	28.881
28.37	58.803	28.713	24.271	28.881
13.42	110.068	37.425	35.996	23.923

31.58	70.217	37.425	34.963	23.923
26.09	78.704	23.207	16.974	22.483
58.62	31.465	37.425	32.176	23.923
77.59	29.064	37.425	33.122	40.058
61.07	47.029	23.155	36.564	23.923
60.79	19.66	21.807	20.402	18.386
33.82	44.11	21.807	41.474	29.334
14.29	38.135	21.807	14.134	22.769
12.35	39.46	38.591	30.459	28.371
43.8	36.973	38.637	36.521	32.912
105.56	45.759	38.636	31.479	34.677
83.87	35.002	27.01	37.207	37.312
4.28	40.514	27.01	24.578	30.709
3.38	38.77	39.261	31.756	52.557
2.94	26.021	27.044	20.37	32.36
6.17	41.57	27.01	34.397	38.413
64.68	20.022	27.919	10.669	21.266
17.62	80.645	41.533	36.369	56.553
33.52	27.223	21.023	14.536	13.094
47.41	34.911	21.023	-9.218	-4.237
68.32	69.827	41.393	37.017	57.57
18.99	58.223	41.579	32.164	30.424
105.69	64.224	21.023	24.179	24.647

	Linear Ensemble			
	Average	Best		Weight
Actual		MMRE		MMRE
8.13	16.45875	SVMreg	20.209	16.5143
13.1	11.24175		8.812	7.47
1.97	39.553		48.164	38.4907
13.49	14.668		14.62	12.8201
11.9	27.98875		24.625	28.2443
7.43	32.42325		28.765	31.9235
10.24	18.83275	MLP	6.999	16.3371
7.87	22.9545		17.064	21.902
16.53	-3.7115		-72.187	-19.1264
12.93	-3.7115		-72.187	-19.1264
18.66	8.1915		14.302	1.7936
30.3	38.15775		27.199	35.1638
18.33	15.28	MLP	-10.493	9.4492
30.82	15.9965		10.502	16.1953

12.58	37.46525		35.34	35.9488
38.67	31.4385		33.814	32.2631
31.25	27.10825		29.531	26.8178
26.22	33.39825		35.365	33.6445
140.13	25.84175	MLP	-1.751	19.7265
6.29	31.3525		38.588	33.1128
5.47	32.84375		28.709	31.1869
23.26	23.6		16.973	21.5232
6.83	29.3805		30.063	29.2823
17.24	25.76525		20.813	24.4363
72.16	34.8955	SVMreg	25.881	32.1104
15.63	41.49725		34.258	40.3914
34.78	41.49725		34.258	40.3914
10.02	31.011		14.527	32.5505
50.96	39.878		24.996	41.4259
35.03	4.823		-0.1	-0.4055
6.56	30.49425	SVMreg	24.384	28.1436
13.26	28.6495		20.493	26.2384
23.92	27.416		26.227	26.4977
21.15	30.14775		15.897	25.4589
36.17	31.49225		17.965	26.6171
28.37	35.167		24.271	29.9788
13.42	51.853	SVMreg	35.996	40.0671
31.58	41.632		34.963	35.6688
26.09	35.342		16.974	26.0463
58.62	31.24725		32.176	30.6788
77.59	34.91725		33.122	35.6576
61.07	32.66775		36.564	31.1364
60.79	20.06375	SVMreg	20.402	19.7893
33.82	34.18125		41.474	36.3925
14.29	24.21125		14.134	22.292
12.35	34.22025		30.459	32.446
43.8	36.26075		36.521	35.7403
105.56	37.63775		31.479	36.0101
83.87	34.13275	SVMreg	37.207	35.5468
4.28	30.70275		24.578	30.8282
3.38	40.586		31.756	38.7709
2.94	26.44875		20.37	25.1307
6.17	35.3475		34.397	36.6134
64.68	19.969		10.669	17.3193
17.62	53.775	SVMreg	36.369	47.8846

33.52	18.969		14.536	16.6695
47.41	10.61975		-9.218	2.7374
68.32	51.45175		37.017	47.3391
18.99	40.5975		32.164	36.1309
105.69	33.51825		24.179	27.6927

	Train set#									
MMRE	1	2	3	4	5	6	7	8	9	10
MLP	1.0426 43	0.8644 56	0.8135 27	0.7867 56	0.9859 14	2.0581 92	2.6677 75	1.4610 55	1.1818 73	1.7672 39
RBF	1.4835 59	1.5711 7	1.6831 14	1.4850 09	1.5793 73	1.6819 89	1.5739 61	1.5198 86	1.3756 06	1.4314 84
SVMreg	0.9304 25	1.0304 98	1.0734 44	1.0574 34	0.9813 1	1.0392 21	0.9897 27	1.2185 01	0.7143 13	1.1803 14
M5P	1.5552 85	1.4401 9	1.9143 39	1.5079 21	1.4348 39	1.7314 28	1.3427 7	1.3643 13	1.2688 09	1.3041 81

	Train set#									
Ranking	1	2	3	4	5	6	7	8	9	10
MLP	3	4	4	4	3	1	1	2	3	1
RBF	2	1	2	2	1	3	2	1	1	2
SVMreg	4	3	3	3	4	4	4	4	4	4
M5P	1	2	1	1	2	2	3	3	2	3

MLP	Weight	RBF	Weight	SVMreg	Weight	M5P	Weight	Total
3.036	3	32.609	2	20.209	4	9.981	1	16.5143
-14.656	3	32.609	2	8.812	4	18.202	1	7.47
24.797	3	32.609	2	48.164	4	52.642	1	38.4907
-3.47	3	32.609	2	14.62	4	14.913	1	12.8201
32.002	3	32.609	2	24.625	4	22.719	1	28.2443
35.319	3	32.609	2	28.765	4	33	1	31.9235
6.999	2	23.713	1	22.424	4	22.195	3	19.3992
17.064	2	23.731	1	24.987	4	26.036	3	23.5915
-72.187	2	23.713	1	6.515	4	27.113	3	-1.3262
-72.187	2	23.713	1	6.515	4	27.113	3	-1.3262
14.302	2	23.713	1	-52.487	4	47.238	3	-1.5917

27.199	2	43.108	1	35.086	4	47.238	3	37.9564
-10.493	3	23.511	1	20.67	4	27.432	2	12.9576
10.502	3	23.511	1	21.475	4	8.498	2	15.7913
35.34	3	42.261	1	30.673	4	41.587	2	35.4147
33.814	3	42.035	1	26.7	4	23.205	2	29.6687
29.531	3	25.536	1	22.808	4	30.558	2	26.6477
35.365	3	42.261	1	27.248	4	28.719		25.7348
-1.751	2	37.779	1	30.686	4	36.653	3	26.698
38.588	2	26.632	1	31.661	4	28.529	3	31.6039
28.709	2	37.659	1	28.354	4	36.653	3	31.8452
16.973	2	27.883	1	21.015	4	28.529	3	23.1476
30.063	2	32.748	1	26.182	4	28.529	3	28.3189
20.813	2	28.575	1	25.144	4	28.529	3	25.6364
30.701	3	40.523	1	25.881	4	42.477	2	32.1104
43.993	3	40.573	1	34.258	4	47.165	2	40.3914
43.993	3	40.573	1	34.258	4	47.165	2	40.3914
70.368	3	22.005	1	14.527	4	17.144	2	32.5505
69.367	3	24.124	1	24.996	4	41.025	2	41.4259
-20.434	3	22.005	1	-0.1	4	17.821	2	-0.4055
39.999	3	28.713	1	24.384	4	28.881	2	30.4008
36.511	3	28.713	1	20.493	4	28.881	2	27.798
35.518	3	28.713	1	26.227	4	19.206	2	27.8587
47.1	3	28.713	1	15.897	4	28.881	2	29.1363
50.41	3	28.713	1	17.965	4	28.881	2	30.9565
58.803	3	28.713	1	24.271	4	28.881	2	35.9968
110.068	1	37.425	2	35.996	4	23.923	3	40.0671
70.217	1	37.425	2	34.963	4	23.923	3	35.6688
78.704	1	23.207	2	16.974	4	22.483	3	26.0463
31.465	1	37.425	2	32.176	4	23.923	3	30.6788
29.064	1	37.425	2	33.122	4	40.058	3	35.6576
47.029	1	23.155	2	36.564	4	23.923	3	31.1364
19.66	2	21.807	1	20.402	4	18.386	3	19.7893
44.11	2	21.807	1	41.474	4	29.334	3	36.3925
38.135	2	21.807	1	14.134	4	22.769	3	22.292
39.46	2	38.591	1	30.459	4	28.371	3	32.446
36.973	2	38.637	1	36.521	4	32.912	3	35.7403
45.759	2	38.636	1	31.479	4	34.677	3	36.0101
35.002	1	27.01	2	37.207	4	37.312	3	34.9786

40.514	1	27.01	2	24.578	4	30.709	3	28.4973
38.77	1	39.261	2	31.756	4	52.557	3	40.1987
26.021	1	27.044	2	20.37	4	32.36	3	25.8669
41.57	1	27.01	2	34.397	4	38.413	3	34.8417
20.022	1	27.919	2	10.669	4	21.266	3	18.2334
80.645	2	41.533	1	36.369	4	56.553	3	51.7958
27.223	2	21.023	1	14.536	4	13.094	3	17.2895
34.911	2	21.023	1	-9.218	4	-4.237	3	4.1262
69.827	2	41.393	1	37.017	4	57.57	3	50.1825
58.223	2	41.579	1	32.164	4	30.424	3	37.7953
64.224	2	21.023	1	24.179	4	24.647	3	32.0128

	Nonlinear models			
Actual	MLP	RBF	SVMreg	M5P
8.13	2.027	31.932	20.066	10.29
13.1	-3.239	31.932	7.27	-6.075
1.97	24.229	32.145	28.586	30.42
13.49	-0.377	31.932	14.408	4.272
11.9	34.603	31.946	29.739	37.085
7.43	38.609	32.287	30.114	40.153
10.24	1.202	25.752	18.469	16.497
7.87	5.448	25.752	29.636	30.377
16.53	-47.379	25.752	-43.094	-55.501
12.93	-47.379	25.752	-43.094	-55.501
18.66	-1.004	25.752	29.346	50.216
30.3	27.732	41.141	36.559	46.855
18.33	-2.72	24.95	6.139	1.589
30.82	14.96	24.95	23.425	20.791
12.58	34.894	40.094	33.626	43.508
38.67	43.872	40.094	32.428	42.112
31.25	28.877	24.95	35.739	38.195
26.22	43.585	40.094	33.31	43.531
140.13	-11.299	37.239	17.504	9.262
6.29	19.884	26.661	36.541	45.411
5.47	19.865	37.239	28.003	36.558
23.26	17.544	26.661	20.758	26.041
6.83	42.848	36.126	29.34	37.772
17.24	16.315	26.662	25.055	29.483
72.16	49.118	42.788	32.356	41.306
15.63	54.989	42.818	40.013	54.395

34.78	54.989	42.818	40.013	54.395
10.02	38.968	20.932	35.516	51.699
50.96	56.905	20.932	47.391	68.381
35.03	-19.122	20.932	-0.964	-13.873
6.56	29.442	27.321	25.679	28.887
13.26	28.053	27.321	22.833	25.845
23.92	27.059	27.321	23.345	24.979
21.15	40.446	27.321	24.429	35.081
36.17	44.005	27.321	26.432	37.968
28.37	53.117	27.322	31.969	45.289
13.42	58.973	39.832	51.399	78.056
31.58	53.968	39.832	29.617	42.861
26.09	73.385	20.959	35.75	49.231
58.62	14.069	39.832	8.274	8.637
77.59	4.265	39.832	28.378	19.127
61.07	38.143	20.959	22.211	22.383
60.79	12.96	20.839	15.537	13.479
33.82	18.741	20.839	33.695	30.252
14.29	18.634	20.839	34.344	20.194
12.35	17.677	41.097	25.846	28.777
43.8	15.225	41.142	27.421	35.734
105.56	22.886	41.179	36.891	38.439
83.87	39.975	27.296	35.483	33.358
4.28	41.948	27.296	26.355	39.479
3.38	100.042	39.196	32.681	37.543
2.94	41.923	27.296	21.6	23.386
6.17	41.116	27.296	34.348	40.652
64.68	42.072	27.296	12.732	16.724
17.62	101.94	43.14	54.875	74.239
33.52	9.954	19.91	12.248	9.647
47.41	15.013	19.91	4.69	-1.823
68.32	82.648	43.139	52.122	69.804
18.99	20.712	42.621	31.525	40.183
105.69	34.128	20.24	30.684	38.054

UIMS (regression) results

Actual	MLP	RBF	SVMreg	M5P
14	13.818	23.515	33.759	36.906
18	22.782	23.515	27.557	17.167
2	8.486	23.515	2.25	-4.419
2	1.155	23.515	-7.535	2.643
10	11.189	136.286	76.679	112.286
16	112.783	22.929	91.079	7.709
16	7.301	22.929	26.404	21.338
18	13.177	22.929	28.471	18.318
2	5.235	22.096	-8.22	1.555
16	29.142	22.096	20.492	20.34
2	0.88	22.096	2.158	-6.704
48	7.117	22.096	27.976	9.931
205	103.066	124.833	141.518	151.921

30	31.534	22.068	33.553	46.394
30	21.333	22.068	8.936	9.053
2	12.192	22.068	2.437	-6.765
12	22.916	20.15	5.834	5.095
50	24.956	20.15	35.326	42.908
26	50.581	153.792	124.292	131.097
39	28.845	20.15	44.49	36.247
15	27.213	19.16	4.325	4.675
119	12.628	19.16	46.214	75.935
2	25.641	19.16	7.187	2.273
18	26.354	19.16	-2.341	3.15
26	18.973	21.813	57.025	74.09
2	3.013	21.813	-9.508	-20.828
2	13.191	21.813	21.627	15.486
48	9.91	21.813	6.108	2.979
34	34.595	22.199	49.877	39.596
93	70.683	138.599	111.486	71.382
2	14.41	22.199	3.073	3.907
168	119.234	138.599	133.37	151.742
30	28.066	20.418	20.488	25.906
17	33.263	146.721	132.98	123.93
27	46.497	20.418	13.354	37.163
30	29.254	20.418	34.436	37.174
253	121.435	101.8	122.82	76.764
192	223.682	101.8	152.223	175.343
20	33.201	22	23.638	29.091

	Ensemble			
	Average	Best		Weight
Actual		MMRE		MMRE
14	26.9995	MLP	13.818	25.7023
18	22.75525		22.782	22.1258
2	7.458		8.486	4.8702
2	4.9445		1.155	2.0994
10	84.11	MLP	11.189	67.1258
16	58.625		112.783	67.9346
16	19.493		7.301	16.8955
18	20.72375		13.177	18.7533
2	5.1665	MLP	5.235	3.1261
16	23.0175		29.142	24.0668

2	4.6075	SVMreg	0.88	0.982
48	16.78		7.117	13.6309
205	130.3345		141.518	130.3945
30	33.38725		33.553	34.367
30	15.3475		8.936	13.9917
2	7.483	MLP	2.437	5.4862
12	13.49875		22.916	13.9506
50	30.835		24.956	31.1768
26	114.9405		50.581	99.1186
39	32.433		28.845	34.1494
15	13.84325	M5P	4.675	9.7208
119	38.48425		75.935	49.333
2	13.56525		2.273	9.4614
18	11.58075		3.15	7.0251
26	42.97525		18.973	41.696
2	-1.3775	MLP	3.013	-3.6315
2	18.02925		13.191	17.043
48	10.2025		9.91	8.5735
34	36.56675		39.596	39.9404
93	98.0375		71.382	89.9951
2	10.89725	M5P	3.907	7.5866
168	135.73625		151.742	138.4145
30	23.7195		20.488	23.622
17	109.2235		132.98	111.6957
27	29.358		13.354	27.8317
30	30.3205	SVMreg	34.436	32.8192
253	105.70475		122.82	111.0913
192	163.262		152.223	173.2424
20	26.9825		23.638	27.4337

	Train set#									
MMRE	1	2	3	4	5	6	7	8	9	10
MLP	0.8644 13	0.6482 96	0.4094 59	1.4093 53	0.9221 47	2.7612 99	0.9238 9	1.7898 9	1.5853 41	0.7652 43
RBF	2.8130 18	3.2779 16	2.6906 67	2.8853 73	2.8749	2.5843 5	2.6546 58	2.9338 53	3.0068 07	3.0898 67
SVMreg	1.2394 26	1.4577 31	1.2636 53	0.8221 22	1.7581 24	1.1994 84	1.0193 67	1.0075 23	1.1303 36	0.7208 13
M5P	1.0917 66	1.0264 88	1.2037 02	1.4904 18	1.8108 03	0.7406 51	1.7640 25	0.7174 08	1.2705 95	1.3878 79

	Train set#									
Ranking	1	2	3	4	5	6	7	8	9	10
MLP	4	4	4	3	4	1	4	2	2	3
RBF	1	1	1	1	1	2	1	1	1	1
SVMreg	2	2	2	4	3	3	3	3	4	4
M5P	3	3	3	2	2	4	2	4	3	2

MLP	Weight	RBF	Weight	SVMreg	Weight	M5P	Weight	Total
13.818	4	23.515	1	33.759	3	36.906	2	25.3876
22.782	4	23.515	1	27.557	3	17.167	2	23.1648
8.486	4	23.515	1	2.25	3	-4.419	2	5.5371
1.155	4	23.515	1	-7.535	3	2.643	2	1.0816
11.189	3	136.286	1	76.679	4	112.286	2	70.1141
112.783	3	22.929	1	91.079	4	7.709	2	74.1012
7.301	3	22.929	1	26.404	4	21.338	2	19.3124
13.177	3	22.929	1	28.471	4	18.318	2	21.298
5.235	4	22.096	1	-8.22	3	1.555	2	2.1486
29.142	4	22.096	1	20.492	3	20.34	2	24.082
0.88	4	22.096	1	2.158	3	-6.704	2	1.8682
7.117	4	22.096	1	27.976	3	9.931	2	15.4354
103.066	4	124.833	1	141.518	3	151.921	2	126.5493
31.534	4	22.068	1	33.553	3	46.394	2	34.1651
21.333	4	22.068	1	8.936	3	9.053	2	15.2314
12.192	4	22.068	1	2.437	3	-6.765	2	6.4617
22.916	4	20.15	1	5.834	3	5.095	2	13.9506
24.956	4	20.15	1	35.326	3	42.908	2	31.1768
50.581	4	153.792	1	124.292	3	131.097	2	99.1186
28.845	4	20.15	1	44.49	3	36.247	2	34.1494
27.213	3	19.16	1	4.325	4	4.675	2	12.7449
12.628	3	19.16	1	46.214	4	75.935	2	39.377
25.641	3	19.16	1	7.187	4	2.273	2	12.9377
26.354	3	19.16	1	-2.341	4	3.15	2	9.5158
18.973	3	21.813	1	57.025	4	74.09	2	45.5012
3.013	3	21.813	1	-9.508	4	-20.828	2	-4.8836
13.191	3	21.813	1	21.627	4	15.486	2	17.8866
9.91	3	21.813	1	6.108	4	2.979	2	8.1933
34.595	2	22.199	1	49.877	3	39.596	4	39.9404
70.683	2	138.599	1	111.486	3	71.382	4	89.9951
14.41	2	22.199	1	3.073	3	3.907	4	7.5866

119.234	2	138.599	1	133.37	3	151.742	4	138.4145
28.066	3	20.418	1	20.488	4	25.906	2	23.838
33.263	3	146.721	1	132.98	4	123.93	2	102.629
46.497	3	20.418	1	13.354	4	37.163	2	28.7651
29.254	3	20.418	1	34.436	4	37.174	2	32.0272
121.435	3	101.8	1	122.82	4	76.764	2	111.0913
223.682	3	101.8	1	152.223	4	175.343	2	173.2424
33.201	3	22	1	23.638	4	29.091	2	27.4337

	Nonlinear models			
Actual	MLP	RBF	SVMreg	M5P
14	13.776	18.733	15.654	14.261
18	18.445	18.732	20.961	23.236
2	6.846	18.732	6.454	8.922
2	3.755	18.732	1.07	1.582
10	18.463	151.537	26.553	13.562
16	106.179	147.229	97.876	114.934
16	5.999	19.169	11.62	9.683
18	10.334	19.169	15.817	15.546
2	3.348	18.05	4.621	10.878
16	22.301	18.05	29.127	34.7
2	-1.998	18.05	1.096	6.539
48	3.135	18.05	11.042	12.753
205	98.767	127.953	107.819	102.466
30	28.93	18.148	29.524	27.718
30	16.694	18.146	12.924	17.059
2	7.67	18.146	2.85	7.507
12	19.297	17.153	16.538	21.359
50	24.928	17.153	24.611	23.452
26	52.831	151.245	54.366	49.744
39	28.98	17.153	27.665	27.443
15	25.135	19.568	8.15	16.258
119	13.326	28.02	33.086	1.025
2	23.756	19.568	6.204	14.616
18	24.381	19.568	6.565	15.361
26	18.322	18.664	28.259	18.267
2	3.777	18.555	-5.701	2.103
2	12.176	18.555	11.518	12.411
48	9.456	18.555	5.063	9.088
34	30.819	17.786	28.675	29.285
93	56.842	148.784	68.367	65.68

2	11.49	17.781	3.122	8.929
168	113.867	148.784	129.405	114.643
30	24.578	17.603	23.432	25.401
17	30.645	143.501	66.132	50.637
27	43.398	17.603	37.388	43.117
30	26.37	17.603	30.018	28.793
253	131.905	120.67	117.969	118.592
192	209.051	120.67	210.249	213.055
20	32.317	18.782	29.461	31.118

QUES (regression) results

Actual	MLP	RBF	SVMreg	M5P
102	280.161	57.846	84.178	68.136
85	81.491	56.883	49.121	51.343
38	60.786	56.883	36.522	36.448
81	76.445	84.991	72.988	73.462
55	105.476	84.991	46.478	59.269
101	230.77	84.991	105.836	130.597
38	42.765	84.991	30.38	42.566
157	116.41	57.649	99.159	88.549
68	89.559	82.615	57.461	76.784
26	82.995	57.649	42.437	63.407
24	48.057	68.29	34.114	41.225
86	125.873	57.649	67.505	64.008
26	51.898	57.649	42.41	53.796
47	52.927	82.037	47	59.516
78	100.279	57.498	56.061	55.433
88	93.322	81.28	64.335	89.383
124	136.317	57.498	73.257	61.976
28	65.041	57.498	55.68	56.35
62	91.68	57.498	49.695	47.677
35	102.332	57.498	54.209	63.208
41	58.99	57.498	51.233	65.21
49	100.731	59.822	83.278	84.617
9	24.628	59.822	5.322	26.3
70	208.525	61.963	136.533	103.448
46	50.458	59.822	87.609	47.316
42	105.19	59.822	83.907	61.088
92	161.97	80.754	98.938	97.664
48	91.174	59.822	74.785	61.734
56	71.903	74.233	66.458	66.868
217	347.932	74.233	147.654	167.422
45	90.006	58.697	78.49	81.067
24	40.463	62.198	32.927	34.977
85	73.511	58.697	55.147	55.205
10	33.261	58.697	20.963	19.031
100	99.609	58.697	65.628	81.52

72	100.146	80.123	77.939	77.964
48	86.887	61.112	57.077	65.937
24	56.519	61.138	13.045	58.008
16	62.499	61.112	27.553	43.371
14	51.453	61.164	27.019	23.565
82	104.478	61.112	56.896	59.825
39	86.021	61.112	48.345	66.093
98	40.939	62.819	78.105	114.787
56	63.519	62.819	86.094	78.992
146	208.11	59.834	196.599	89.101
25	45.496	62.819	70.42	90.963
68	87.483	62.819	57.152	63.52
48	42.483	62.819	40.018	53.793
170	60.974	59.834	79.732	96.405
80	87.311	87.24	101.988	88.553
148	122.701	56.818	71.753	84.416
30	49.75	56.818	36.263	50.31
28	22.48	57.098	23.928	20.411
35	20.881	87.24	18.829	11.069
77	105.704	56.818	61.728	78.67
45	71.64	87.24	56.758	58.712
52	69.697	54.851	45.165	51.707
70	146.683	82.487	60.454	51.076
188	157.792	54.851	84.456	64.692
79	99.475	54.851	60.474	61.097
30	49.174	54.851	36.961	52.518
75	103.625	54.851	59.7	56.402
64	71.033	54.851	45.313	52.881
107	128.705	59.027	68.712	57.328
8	12.922	59.027	19.311	24.307
6	13.934	59.027	18.579	20.447
24	37.902	59.027	54.909	62.63
52	26.061	59.027	32.508	42.504
38	0.649	88.264	44.209	73.485
41	30.853	88.264	46.098	65.019
94	58.643	59.027	54.031	64.982

	Ensemble			
	Average	Best		Weight
Actual		MMRE		MMRE
102	122.58025	SVMreg	84.178	115.9288

85	59.7095		49.121	57.0378
38	47.65975		36.522	43.3887
81	76.9715		72.988	75.0219
55	74.0535		46.478	65.9662
101	138.0485		105.836	136.1666
38	50.1755		30.38	41.9739
157	90.44175	SVMreg	99.159	95.2752
68	76.60475		57.461	72.1929
26	61.622		42.437	58.3608
24	47.9215		34.114	42.4535
86	78.75875		67.505	77.1439
26	51.43825		42.41	49.2473
47	60.37	SVMreg	47	55.4439
78	67.31775		56.061	64.8599
88	82.08		64.335	79.3413
124	82.262		73.257	80.9088
28	58.64225		55.68	57.935
62	61.6375		49.695	58.2669
35	69.31175	SVMreg	54.209	66.8622
41	58.23275		51.233	57.604
49	82.112		83.278	84.8247
9	29.018		5.322	20.9266
70	127.61725		136.533	133.5489
46	61.30125		87.609	65.3122
42	77.50175	M5P	83.907	78.9094
92	109.8315		98.938	109.3438
48	71.87875		74.785	72.6512
56	69.8655		66.868	68.4885
217	184.31025		167.422	188.2747
45	77.065		81.067	79.8447
24	42.64125	SVMreg	34.977	38.1813
85	60.64		55.205	59.198
10	32.988		19.031	26.4232
100	76.3635		81.52	78.0879
72	84.043		77.939	82.6063
48	67.75325		57.077	66.1005
24	47.1775		13.045	40.038
16	48.63375		27.553	42.6435
14	40.80025		27.019	34.2841
82	70.57775		56.896	67.7127
39	65.39275		48.345	62.4813

98	74.1625	SVMreg	78.105	80.1478
56	72.856		86.094	77.1209
146	138.411		196.599	152.9753
25	67.4245		70.42	70.838
68	67.7435		57.152	65.6953
48	49.77825		40.018	46.9236
170	74.23625		79.732	78.9925
80	91.273	SVMreg	101.988	93.5473
148	83.922		71.753	84.248
30	48.28525		36.263	45.23
28	30.97925		23.928	25.9003
35	34.50475		18.829	23.7525
77	75.73		61.728	75.1148
45	68.5875		56.758	63.3688
52	55.355	SVMreg	45.165	54.8016
70	85.175		60.454	86.6504
188	90.44775		84.456	99.5435
79	68.97425		60.474	71.7366
30	48.376		36.961	45.5253
75	68.6445		59.7	71.733
64	56.0195		45.313	55.4964
107	78.443	MLP	128.705	89.4639
8	28.89175		12.922	21.7262
6	27.99675		13.934	21.1394
24	53.617		37.902	50.0622
52	40.025		26.061	34.5803
38	51.65175		0.649	37.0457
41	57.5585		30.853	48.0008
94	59.17075		58.643	58.5656

	Train set#									
MMRE	1	2	3	4	5	6	7	8	9	10
MLP	0.5997 94	0.8506 51	0.9067 63	0.4945 02	0.5583 18	0.7206 68	0.5831 78	0.3592 47	0.3430 78	0.2621 09
RBF	0.9711 21	0.9445 32	0.9729 06	0.9500 04	0.8950 87	0.9503 55	1.0024 43	0.9429 68	0.9377 54	0.7498 11
SVMreg	0.2893 25	0.3697 04	0.3567 11	0.3050 58	0.3007 35	0.3340 08	0.2467 24	0.2845 06	0.3113 28	0.3300 26
M5P	0.3266 71	0.5340 99	0.5139 07	0.3557 17	0.2702 93	0.5675 17	0.2650 12	0.3149 07	0.5053 92	0.5119 83

	Train set#									
--	------------	--	--	--	--	--	--	--	--	--

Ranking	1	2	3	4	5	6	7	8	9	10
MLP	2	2	2	2	2	2	2	2	3	4
RBF	1	1	1	1	1	1	1	1	1	1
SVMreg	4	4	4	4	3	4	4	4	4	3
M5P	3	3	3	3	4	3	3	3	2	2

MLP	Weight	RBF	Weight	SVMreg	Weight	M5P	Weight	Total
280.161	2	57.846	1	84.178	4	68.136	3	115.9288
81.491	2	56.883	1	49.121	4	51.343	3	57.0378
60.786	2	56.883	1	36.522	4	36.448	3	43.3887
76.445	2	84.991	1	72.988	4	73.462	3	75.0219
105.476	2	84.991	1	46.478	4	59.269	3	65.9662
230.77	2	84.991	1	105.836	4	130.597	3	136.1666
42.765	2	84.991	1	30.38	4	42.566	3	41.9739
116.41	3	57.649	2	99.159	4	88.549	1	94.9713
89.559	3	82.615	2	57.461	4	76.784	1	74.0535
82.995	3	57.649	2	42.437	4	63.407	1	59.7438
48.057	3	68.29	2	34.114	4	41.225	1	45.8432
125.873	3	57.649	2	67.505	4	64.008	1	82.6945
51.898	3	57.649	2	42.41	4	53.796	1	49.4428
52.927	3	82.037	2	47	4	59.516	1	57.0371
100.279	1	57.498	3	56.061	4	55.433	2	60.7883
93.322	1	81.28	3	64.335	4	89.383	2	77.3268
136.317	1	57.498	3	73.257	4	61.976	2	72.5791
65.041	1	57.498	3	55.68	4	56.35	2	57.2955
91.68	1	57.498	3	49.695	4	47.677	2	55.8308
102.332	1	57.498	3	54.209	4	63.208	2	61.8078
58.99	1	57.498	3	51.233	4	65.21	2	56.6836
100.731	3	59.822	1	83.278	4	84.617	2	86.4361
24.628	3	59.822	1	5.322	4	26.3	2	20.7594
208.525	3	61.963	1	136.533	4	103.448	2	144.0566
50.458	3	59.822	1	87.609	4	47.316	2	65.6264
105.19	3	59.822	1	83.907	4	61.088	2	83.3196
161.97	3	80.754	1	98.938	4	97.664	2	115.7744
91.174	3	59.822	1	74.785	4	61.734	2	75.5952
71.903	2	74.233	1	66.458	3	66.868	4	68.4885
347.932	2	74.233	1	147.654	3	167.422	4	188.2747
90.006	2	58.697	1	78.49	3	81.067	4	79.8447
40.463	2	62.198	1	32.927	3	34.977	4	38.1813
73.511	2	58.697	1	55.147	3	55.205	4	59.198

33.261	2	58.697	1	20.963	3	19.031	4	26.4232
99.609	2	58.697	1	65.628	3	81.52	4	78.0879
100.146	3	80.123	2	77.939	4	77.964	1	85.0404
86.887	3	61.112	2	57.077	4	65.937	1	67.713
56.519	3	61.138	2	13.045	4	58.008	1	40.2021
62.499	3	61.112	2	27.553	4	43.371	1	46.3304
51.453	3	61.164	2	27.019	4	23.565	1	40.8328
104.478	3	61.112	2	56.896	4	59.825	1	72.3067
86.021	3	61.112	2	48.345	4	66.093	1	63.976
40.939	2	62.819	1	78.105	4	114.787	3	80.1478
63.519	2	62.819	1	86.094	4	78.992	3	77.1209
208.11	2	59.834	1	196.599	4	89.101	3	152.9753
45.496	2	62.819	1	70.42	4	90.963	3	70.838
87.483	2	62.819	1	57.152	4	63.52	3	65.6953
42.483	2	62.819	1	40.018	4	53.793	3	46.9236
60.974	2	59.834	1	79.732	4	96.405	3	78.9925
87.311	4	87.24	1	101.988	3	88.553	2	91.9554
122.701	4	56.818	1	71.753	3	84.416	2	93.1713
49.75	4	56.818	1	36.263	3	50.31	2	46.5227
22.48	4	57.098	1	23.928	3	20.411	2	25.9624
20.881	4	87.24	1	18.829	3	11.069	2	24.9389
105.704	4	56.818	1	61.728	3	78.67	2	82.2158
71.64	4	87.24	1	56.758	3	58.712	2	66.1498
69.697	4	54.851	2	45.165	3	51.707	1	57.5692
146.683	4	82.487	2	60.454	3	51.076	1	98.4144
157.792	4	54.851	2	84.456	3	64.692	1	105.893
99.475	4	54.851	2	60.474	3	61.097	1	75.0121
49.174	4	54.851	2	36.961	3	52.518	1	46.9799
103.625	4	54.851	2	59.7	3	56.402	1	75.9704
71.033	4	54.851	2	45.313	3	52.881	1	58.2654
128.705	4	59.027	2	68.712	3	57.328	1	89.6338
12.922	4	59.027	2	19.311	3	24.307	1	25.1982
13.934	4	59.027	2	18.579	3	20.447	1	24.9974
37.902	4	59.027	2	54.909	3	62.63	1	49.7019
26.061	4	59.027	2	32.508	3	42.504	1	36.2326
0.649	4	88.264	2	44.209	3	73.485	1	38.5236
30.853	4	88.264	2	46.098	3	65.019	1	50.3253
58.643	4	59.027	2	54.031	3	64.982	1	57.9701

Actual	Nonlinear models			
	MLP	RBF	SVMreg	M5P

102	237.308	56.001	223.747	245.007
85	72.161	56.001	56.913	61.702
38	52.847	56.001	36.158	39.795
81	73.192	90.299	64.308	61.854
55	96.647	88.134	82.896	85.09
101	222.674	92.948	203.348	213.646
38	45.069	85.139	27.551	24.798
157	101.127	58.36	89.448	89.182
68	78.906	80.847	67.854	73.498
26	63.32	57.825	48.456	53.852
24	41.409	80.847	23.929	22.328
86	116.71	57.848	85.328	99.187
26	41.723	57.826	25.879	20.973
47	53.198	80.847	37.683	34.473
78	91.675	56.43	62.209	72.234
88	90.529	86.575	70.556	70.145
124	131.252	56.43	93.697	109.97
28	61.982	56.43	36.766	36.532
62	82.949	56.43	54.063	51.232
35	94.375	56.43	62.908	74.229
41	59.607	56.43	30.758	31.5
49	93.213	58.975	83.746	87.387
9	22.353	58.975	8.561	11.043
70	209.583	58.975	174.767	183.524
46	41.517	58.975	44.453	37.184
42	97.68	58.975	85.075	86.79
92	158.06	85.827	138.564	142.446
48	82.54	58.975	73.185	74.872
56	66.589	90.378	60.416	57.075
217	196.079	100.965	287.892	306.162
45	75.656	53.596	72.13	75.304
24	35.231	51.399	25.317	23.617
85	61.216	51.165	53.518	55.477
10	25.749	50.761	14.886	13.802
100	86.095	52.343	79.14	83.212
72	90.815	76.373	80.852	82.022
48	76.43	62.09	58.453	61.646
24	52.098	62.099	26.101	31.131
16	53.905	62.092	31.862	37.132
14	42.758	62.126	20.876	26.048
82	94.291	62.09	72.411	79.328

39	75.418	62.089	56.454	60.775
98	59.243	61.501	55.574	44.53
56	61.396	61.436	57.844	54.44
146	221.501	62.267	164.16	178.119
25	56.43	61.444	49.373	42.353
68	75.218	61.423	63.705	70.624
48	44.448	61.439	31.591	30.476
170	67.731	62.075	65.613	56.689
80	86.543	95.863	86.141	78.606
148	124.397	55.76	101.471	114.3
30	55.484	55.76	36.96	41.534
28	30.353	55.76	11.877	9.118
35	43.099	86.389	20.066	13.096
77	108.9	55.76	86.441	98.485
45	75.753	88.421	67.649	66.99
52	64.766	53.579	57.658	61.564
70	147.663	90.3	140.223	145.877
188	149.016	53.579	139.862	147.393
79	100.45	53.579	85.28	90.576
30	44.632	53.579	38.326	41.568
75	104.318	53.579	89.284	94.619
64	66.605	53.579	58.796	62.865
107	125.595	55.344	115.85	120.812
8	6.317	55.564	13.817	14.657
6	7.594	56.054	14.008	15.333
24	36.414	54.86	44.428	42.952
52	22.103	54.883	28.564	28.344
38	18.518	74.476	14.64	9.919
41	43.815	69.362	37.096	35.234
94	52.605	54.889	60.31	59.865

VITA

Personal information

- Hamoud Ibrahim Hamad Aljamaan
- Saudi nationality
- Born in Kuwait, 1985
- Married

Education

- Masters degree in Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. Expected graduation date, June, 2009.
- Bachelor's degree in Computer Science, King Saud University (KSU), Riyadh, Saudi Arabia, June 2006.

Research and Publications

- Hamoud Aljamaan, Mahmoud Elish, "An Empirical Study of Bagging and Boosting Ensembles for Identifying Faulty Classes in Object-Oriented Software", IEEE Symposium on Computational Intelligence and Data Mining (CIDM), March 2009, (Paper was presented by me).
- A. Bahjat, H. Aljamaan, M. Alshayeb; "SQL-GUARD DESIGN PATTERN". SEDE-2009 - 18th International Conference on Software Engineering and Data Engineering, USA, June 2009.

Professional Experience

- 2006, November – Present, graduate assistant (GA), department ICS, KFUPM.
- 2005 summer, developed and enhanced an image processing program, Integrated Solution Service Department (ISSD) of Saudi Aramco, Dhahran, Saudi Arabia.

Research Interests

- Software quality assessment
- Utilization of data mining in software engineering
- Empirical software engineering
- Software project management
- Software privacy and protection